НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок для пошуку репетиторів

Виконав студент IV курсу, групи ІТ-02
(шифр групи)
Скачкова Анастасія Дмитрівна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент Лісовиченко О. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент професор, д.т.н., професор каф. ІСТ Корнага Я.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Скачкової Анастасії Дмитрівни
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для пошуку репетиторів.
керівник проєкту Лісовиченко Олег Іванович, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27» травня 2024 р. № 2112-с

2. Термін подання студентом проєкту«17» червня2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____

2) Схема бази даних _____

3) Креслення вигляду екранних форм _____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «11» березня 2024 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Аналіз існуючих методів розв’язання задачі	26.03.2024	
3	Постановка та формалізація задачі	02.04.2024	
4	Розробка інформаційного забезпечення	10.04.2024	
5	Алгоритмізація задачі	17.04.2024	
6	Обґрунтування вибору використаних технічних засобів	20.04.2024	
7	Розробка програмного забезпечення	11.05.2024	
8	Налагодження програми	18.05.2024	
9	Виконання графічних документів	24.05.2024	
10	Оформлення пояснювальної записки	31.05.2024	
11	Подання ДП на попередній захист	02.06.2024	
12	Подання ДП рецензенту	10.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент _____ Анастасія СКАЧКОВА
(підпис) (ініціали, прізвище)

Керівник _____ Олег ЛІСОВИЧЕНКО
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 100 таблиць, 36 рисунків та 9 джерел – загалом 109 сторінок.

Дипломний проєкт присвячений створенню вебзастосунку, що забезпечить у собі можливість виставлення завдань репетитором для учня та пошук викладачів.

Мета – підвищити ефективність пошуку та підбору розкладу занять з репетитором.

Об'єкт дослідження: технології та методи реалізації вебзастосунку, база даних.

Предмет дослідження: протоколи роботи з базою даних, авторизації, аутентифікації, інтерфейсна реалізація.

У розділі передпроектного обстеження предметної області описано предметну область, проаналізовано існуючі рішення, відомі технічні рішення, описано бізнес-процеси, виконана постановка задачі.

У розділі розроблення вимог до програмного забезпечення визначено варіанти використання програмного забезпечення, проаналізовано системні вимоги, розроблено функціональні та нефункціональні вимоги.

У розділі конструювання та розроблення програмного забезпечення виконано моделювання програмного забезпечення; обґрунтовано вибір архітектури; описане конструювання програмного забезпечення і проведений аналіз безпеки даних.

У розділі аналізу якості та тестування програмного забезпечення розглянуто метрики якості коду, описані процеси тестування; наведений опис контрольного прикладу.

У розділі розгортання та супроводу програмного забезпечення описано процес розгортання і зазначено спосіб його підтримки.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, ASP.NET CORE, CQRS, MVC, CLEAN ARCHITECTURE, MYSQL.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 100 tables, 36 figures and 9 sources – in total 109 pages.

The purpose of the diploma project is to develop a web application that enables tutors to post assignments for students and facilitates the search for instructors.

The aim of the work is to increase the efficiency of searching for and scheduling tutoring sessions.

The research object is the technologies and methods of implementing the web application, as well as the database.

The Pre-project Survey section describes the subject area, analyzes existing solutions and known technical solutions, describes business processes, and defines the task.

The section on Developing Software Requirements identifies software use cases, analyzes system requirements, and develops functional and non-functional requirements.

The Construction and Development of Software section involves modeling software, justifying the choice of architecture, describing software construction, and conducting a data security analysis.

The Quality Analysis and Testing of Software section covers code quality metrics, describes testing processes, and provides a description of a control example.

The Deployment and Maintenance of Software section describes the deployment process and specifies the method of its support.

KEYWORDS: WEB APPLICATION, ASP.NET CORE, CQRS, MVC, CLEAN ARCHITECTURE, MYSQL.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-0221.045440.01.91	Технічне завдання	24	
3	А4	КПІ.ІТ-0221.045440.02.81	Пояснювальна записка	109	
4	А4	КПІ.ІТ-0221.045440.03.12	Текст програми	129	
5	А4	КПІ.ІТ-0221.045440.04.51	Програма та методика тестування	7	
6	А4	КПІ.ІТ-0221.045440.05.34	Керівництво користувача	27	
7	А3	КПІ.ІТ-0221.045440.06.99	Графічний матеріал	3	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-0221.045440.00.90		
Розроб.		Скачкова А. Д.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Лісовиченко О.І.					
Н.контр.		Ліщук К.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-02	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ПОШУКУ РЕПЕТИТОРІВ

Технічне завдання

КПІ.ІТ-0221.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Анастасія СКАЧКОВА

Київ – 2024

ЗМІСТ

НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебзастосунок для пошуку репетиторів.

Галузь застосування: освіта.

Наведене технічне завдання поширюється на розробку програмного забезпечення «Вебзастосунок для пошуку репетиторів» TutorSearch [IT-0221.045440.01.91], котра використовується для пошуку репетиторів та призначена для освітньої галузі, спрямована на учнів та репетиторів.

ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки вебзастосунку є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для пошуку викладачів та виставлення завдань репетитором учню на одній платформі.

Метою розробки є підвищення ефективності пошуку та підбору розкладу занять з репетитором.

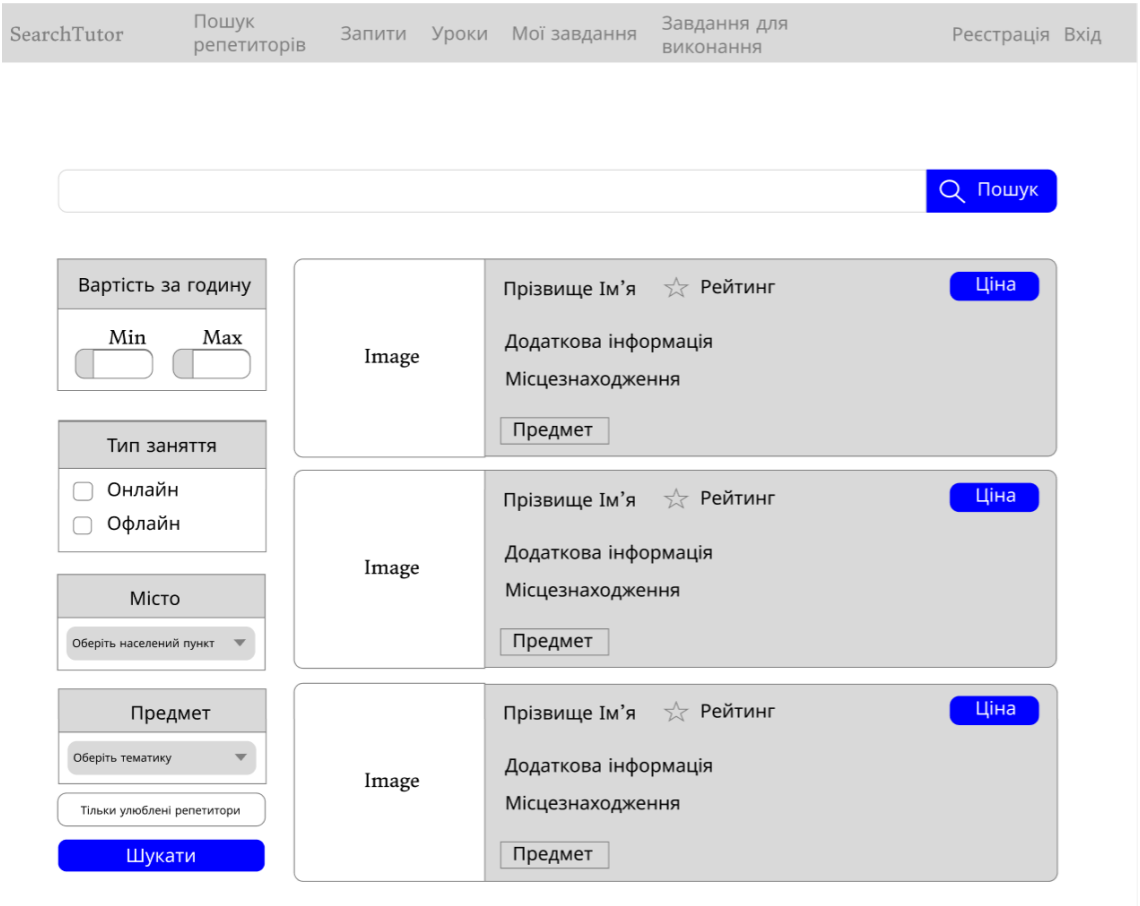
ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

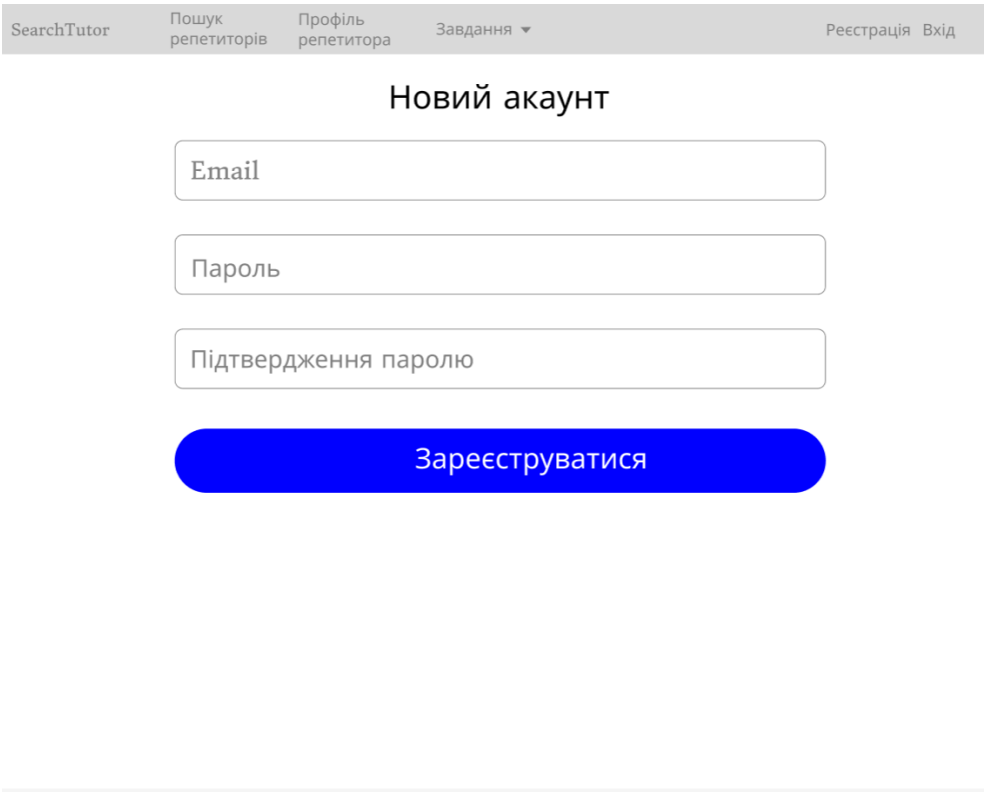
Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

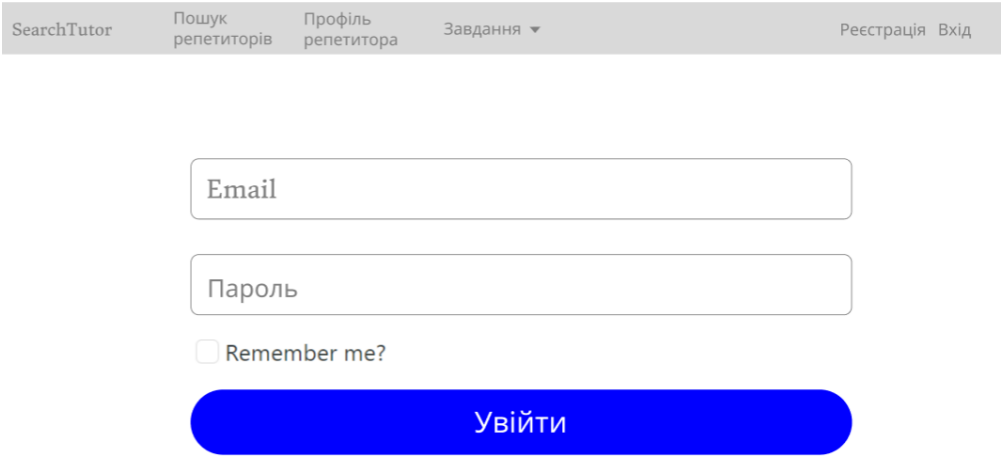
– сторінка пошуку репетиторів та фільтрації за ціною, типом заняття, містом та предметом;



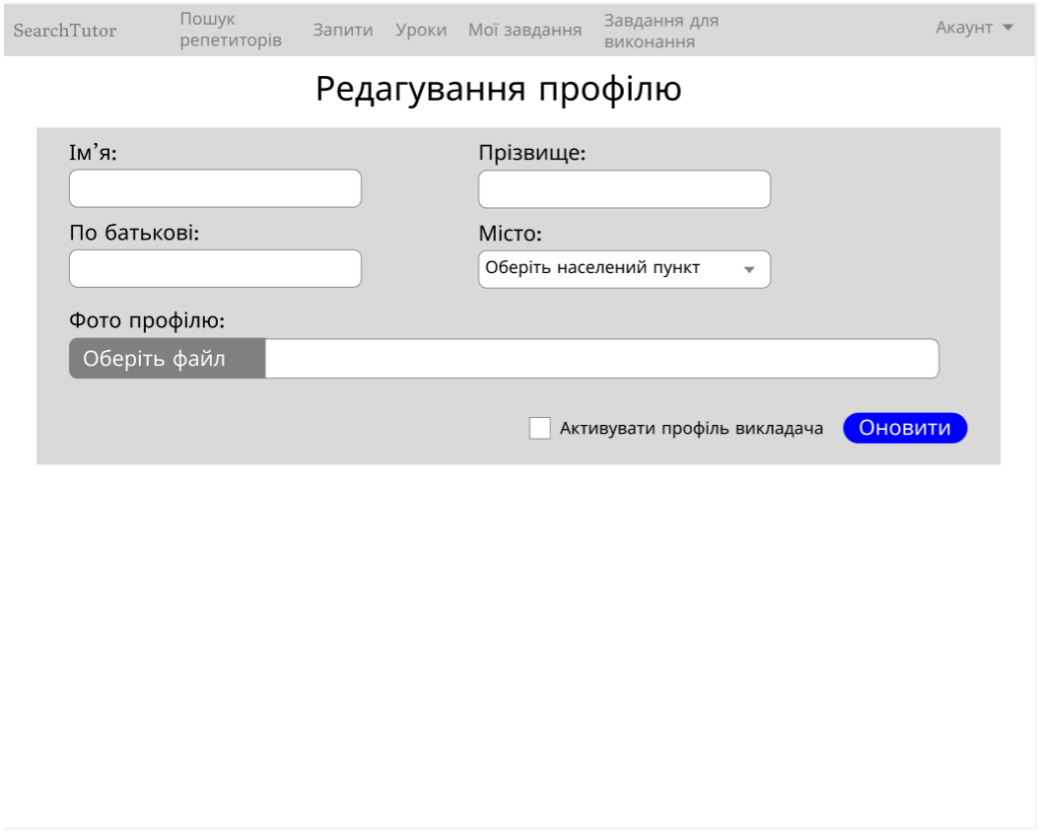
– сторінка реєстрації;



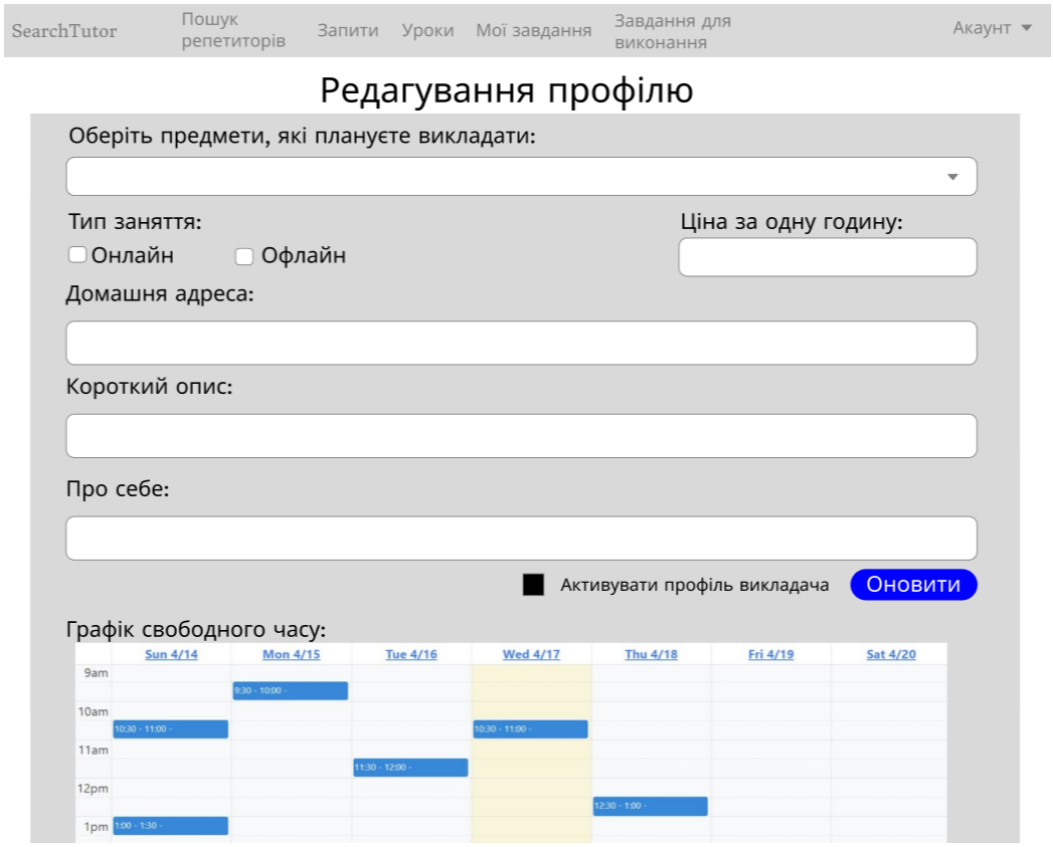
– сторінка входу;



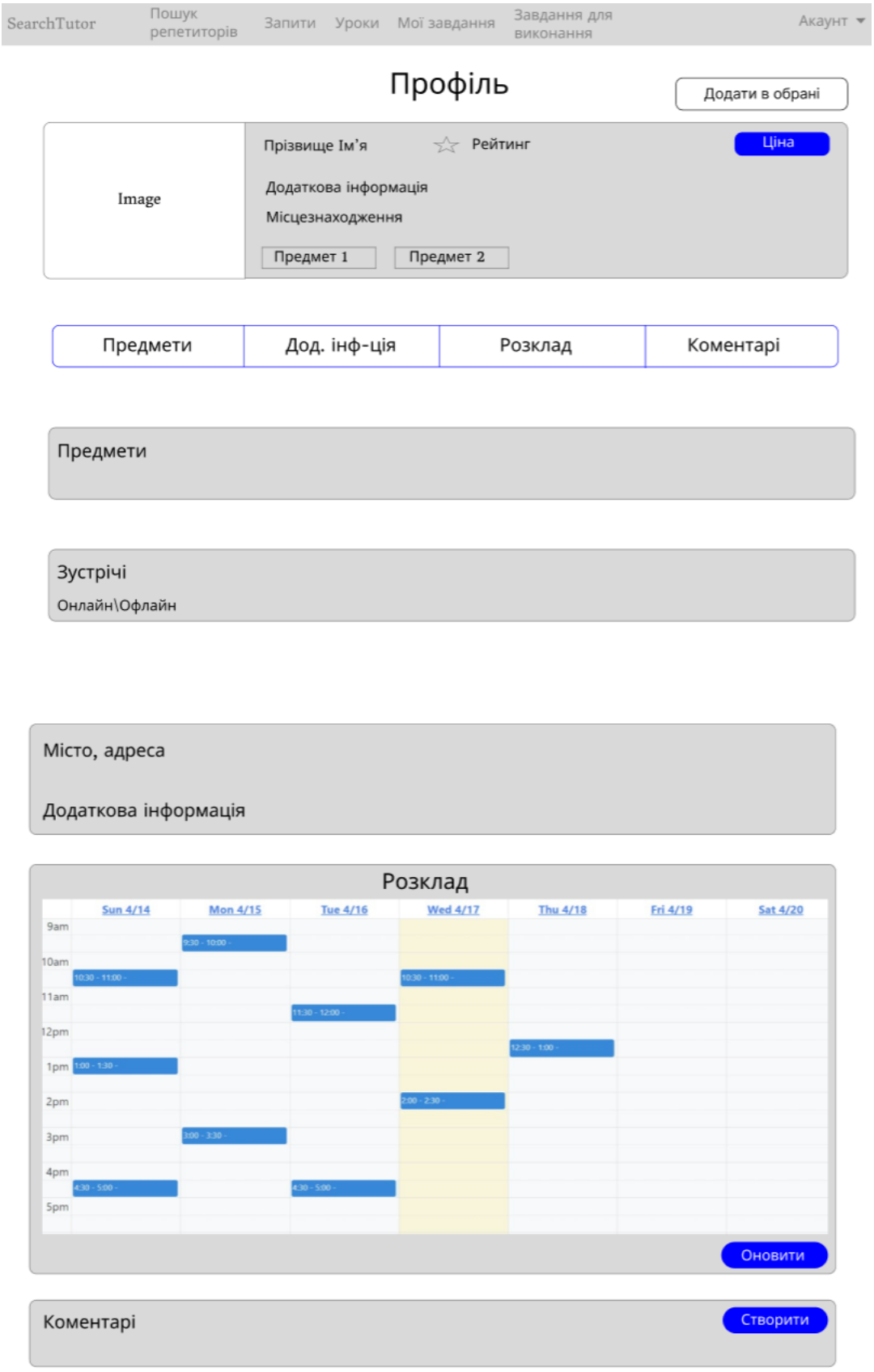
– сторінка створення профілю користувача;



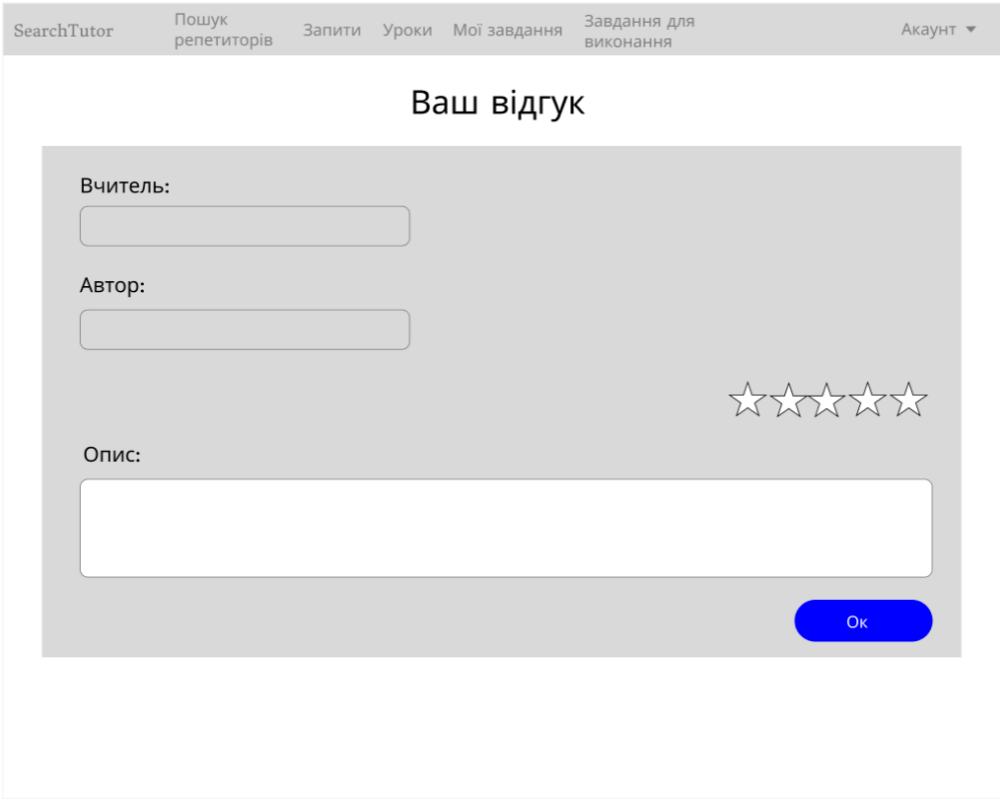
– сторінка створення профілю викладача;



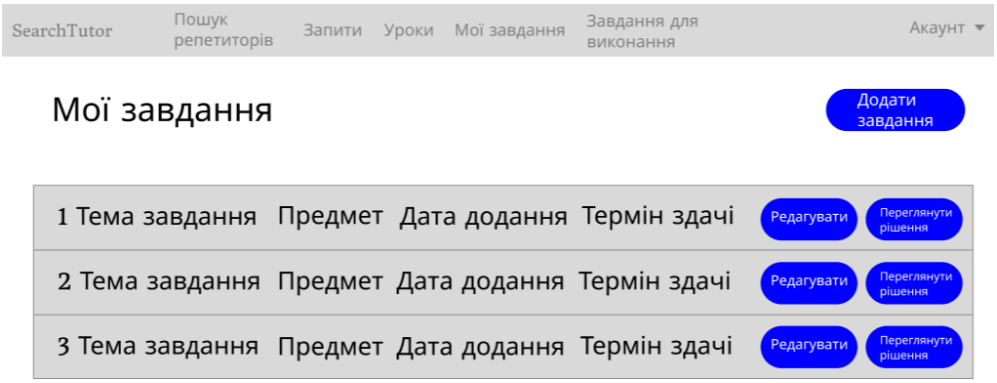
– сторінка профілю;



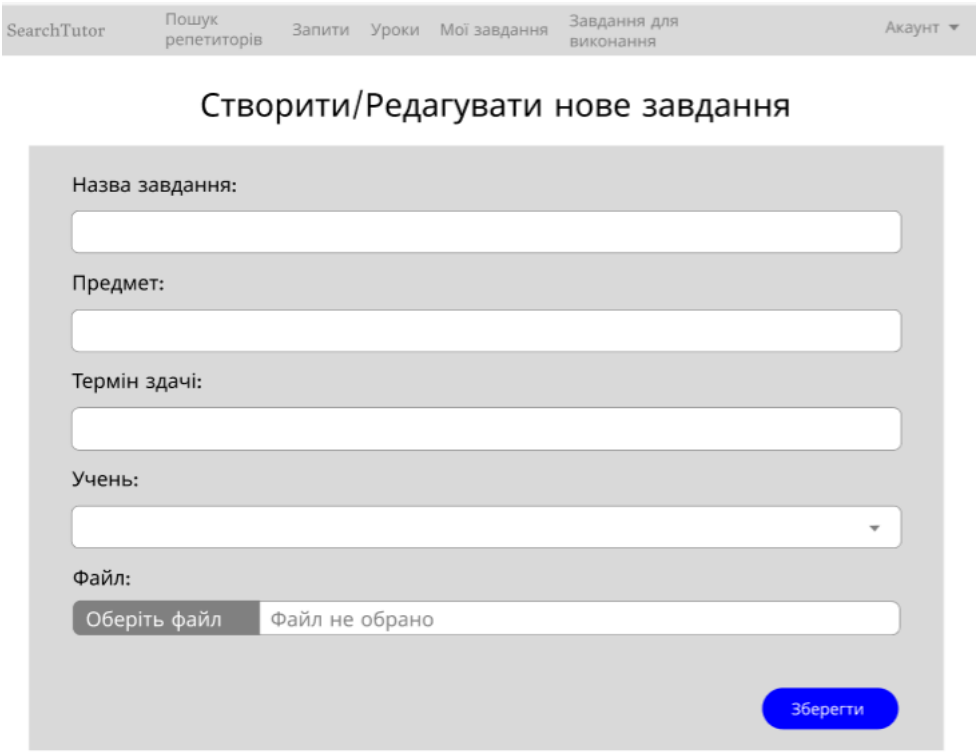
– сторінка створення коментаря до викладача;



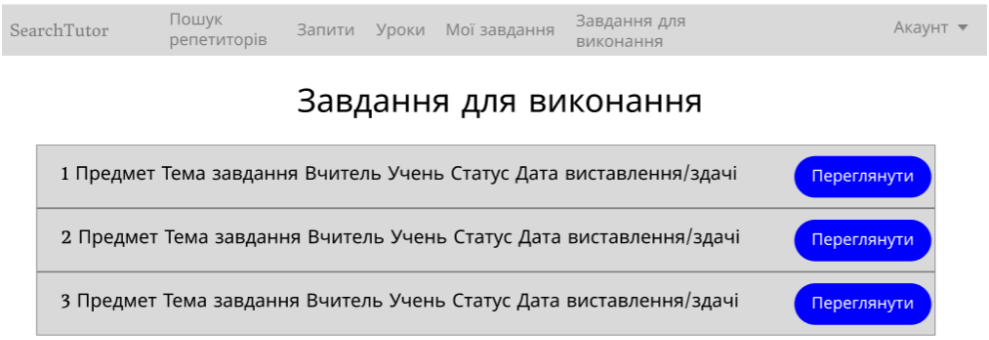
– сторінка завдань зі сторони репетитора;



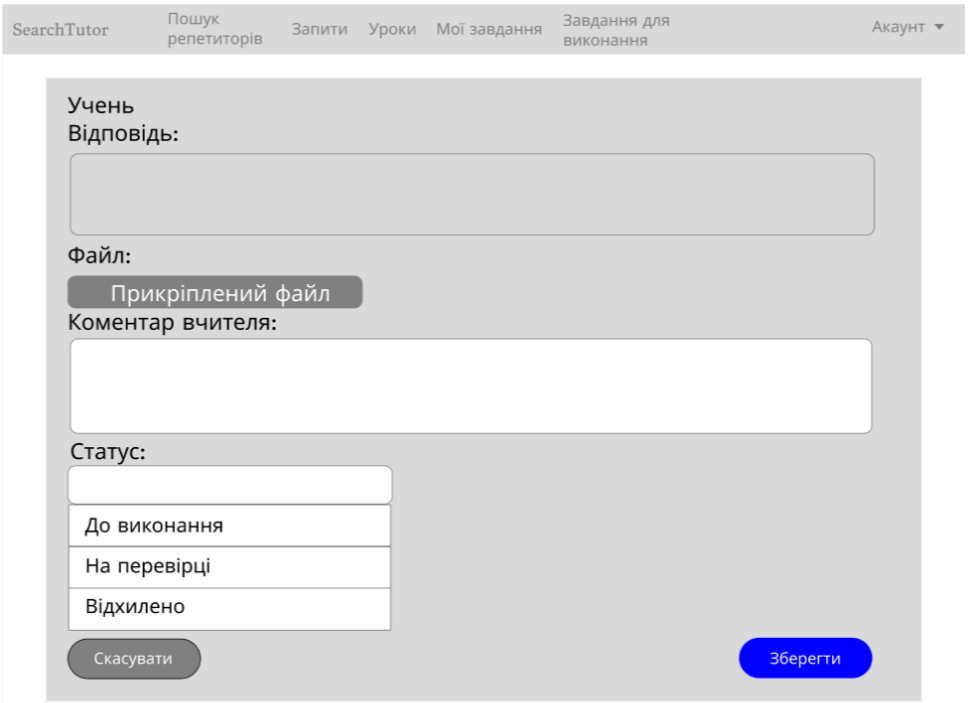
– сторінка створення/редагування завдання;



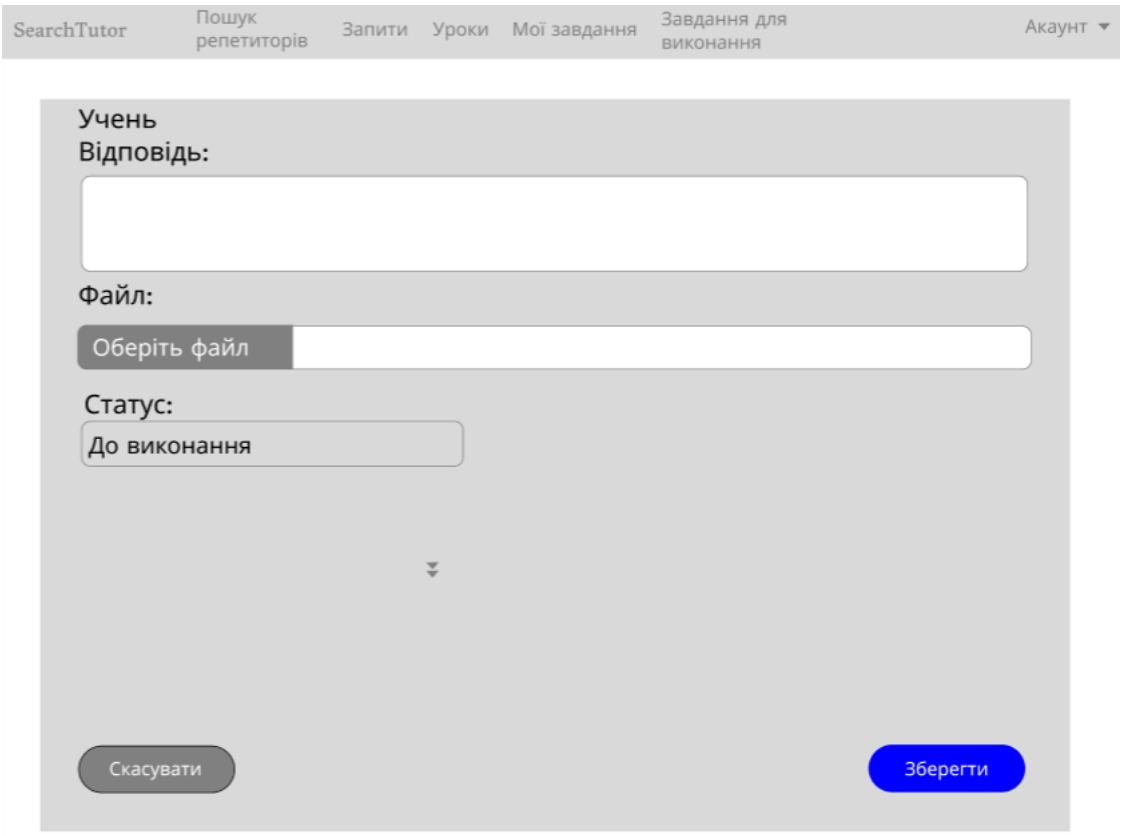
– сторінка перегляду виставлених завдань;



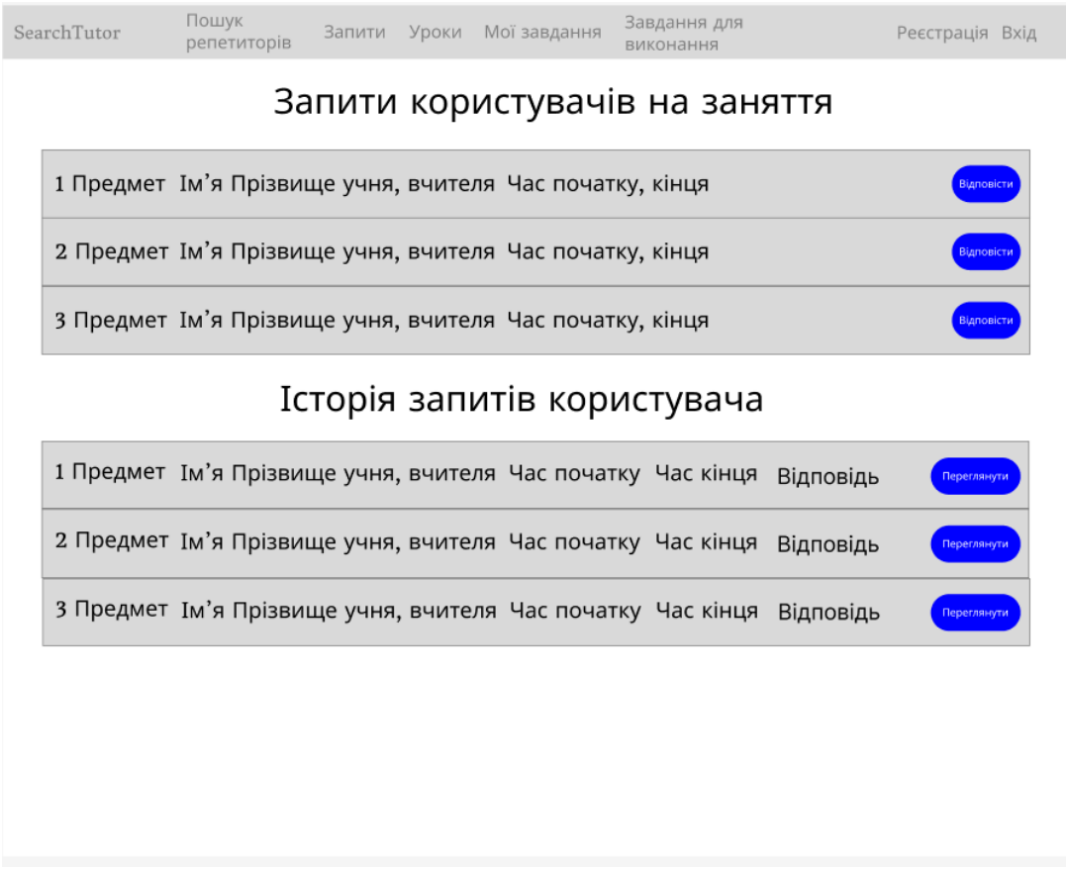
– сторінка перегляду відповіді викладачем;



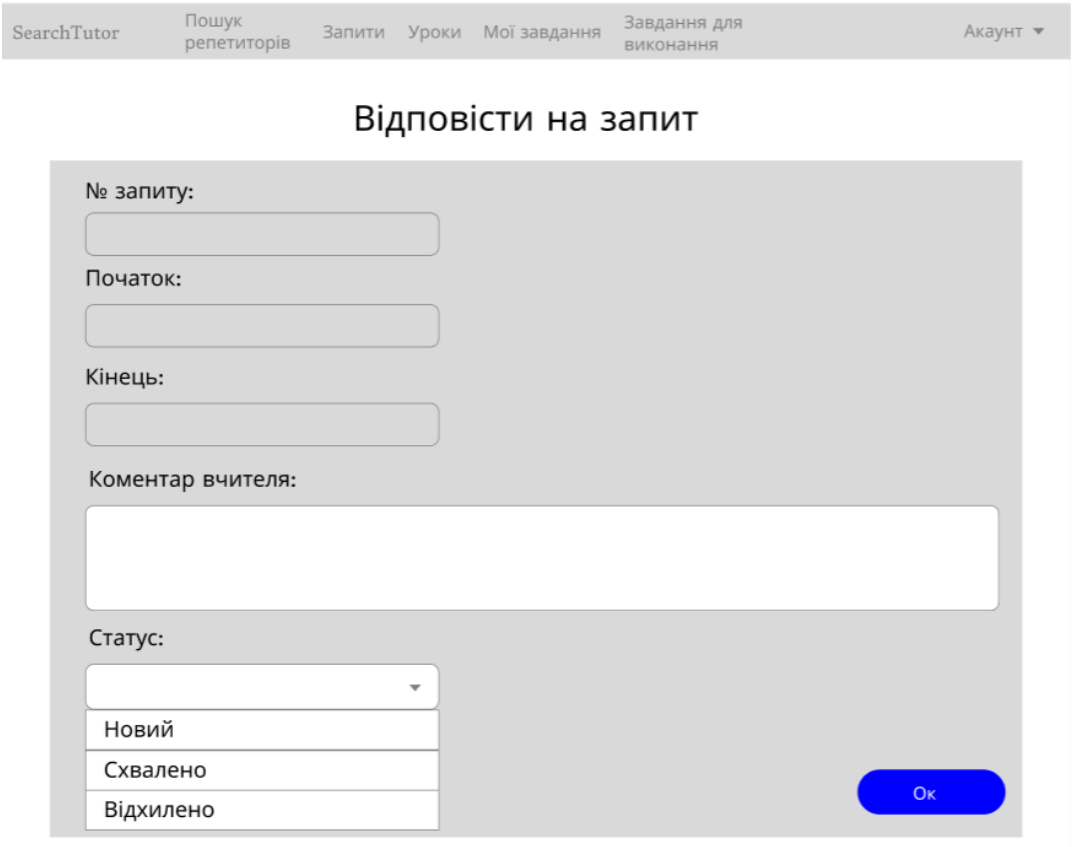
– сторінка створення відповіді;



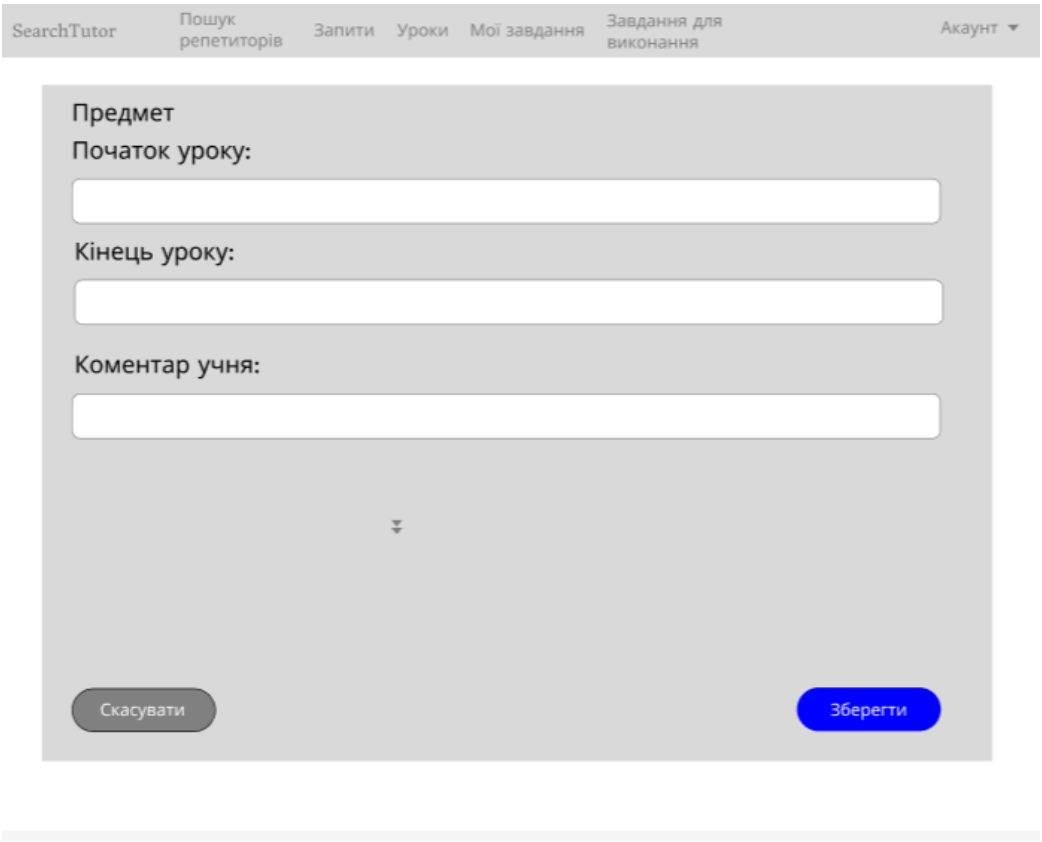
– сторінка запитів користувачів;



– сторінка відповіді на запит;



– форма запису до репетитора;



4.1.2 Для користувача:

1) створення акаунту користувача:
   а) поле "Email":
      – максимальна довжина поля: 254 символи;
      – повинно мати формат електронної пошти;
      – обов’язкове для заповнення.
   б) поле "Пароль":
      – обов'язкове для заповнення;
      – пароль повинен містити принаймні 6 символів;
      – введені символи в цьому полі мають бути приховані.
   в) поле "Підтвердження паролю":
      – введені символи в цьому полі мають бути приховані;
      – повинно співпадати з полем "Пароль";
      – обов'язкове для заповнення.

2) вхід в акаунт:

а) поле "Email":

для входу в акаунт потрібно ввести в це поле електронну пошту, яку користувач вказав при реєстрації у системі.

б) поле "Пароль":

для входу в акаунт потрібно ввести в це поле пароль, який користувач вказав при реєстрації у системі.

в) чек-бокс “Remember me?”:

після вибору цього поля під час авторизації та успішного проходження необхідно, щоб користувач залишався авторизованим у системі при подальших відвідуваннях сайту з того самого пристрою.

3) заповнення профілю користувача:

а) поле «Ім’я»:

– обов’язкове для заповнення під час створення профілю;
– максимальна довжина поля: 50 символів.

б) кнопка «Фото профілю»:

– дозволяє завантажити файл зі свого пристрою;
– може залишитись пустим.

в) поле «Прізвище»:

– обов’язкове для заповнення під час створення профілю;
– максимальна довжина поля: 50 символів.

г) поле «По батькові»:

– обов’язкове для заповнення під час створення профілю;
– максимальна довжина поля: 50 символів.

ґ) випадаючий список «Місто»:

обов’язкове для вибору під час створення профілю.

д) чек-бокс «Активувати профіль репетитора»:

має бути деактивований за замовчуванням.

е) кнопка «Оновити»:

дозволяє зберегти інформацію.

4) заповнення профілю репетитора:

а) поле «Про себе»:

– обов’язкове для заповнення під час створення профілю;

– максимальна довжина поля: 3000 символів;

– мінімальна довжина поля: 50 символів.

б) поле «Домашня адреса»:

– обов’язкове для заповнення під час створення профілю;

– максимальна довжина поля: 300 символів.

в) чек-бокс «Онлайн»:

має бути деактивований за замовчуванням.

г) чек-бокс «Офлайн»:

має бути деактивований за замовчуванням.

ґ) поле «Короткий опис»:

– обов’язкове для заповнення під час створення профілю;

– максимальна довжина поля: 254 символи.

д) поле «Ціна за 1 годину»:

– обов’язкове для заповнення під час створення профілю;

– Значення повинно бути в діапазоні від 10 до 9999.

е) випадаючий список «Предмети»:

– має бути не обрано за замовчуванням;

– дозволяє обрати декілька предметів.

є) кнопка «Оновити»:

дозволяє зберегти інформацію.

5) створення завдання:

а) поле «Назва завдання»:

– максимальна довжина поля: 50 символів;

– мінімальна довжина поля: 3 символи;

– обов’язкове для заповнення під час створення завдання.

б) поле «Опис завдання»:

– максимальна довжина поля: 500 символів;

– може залишитись пустим.

в) поле «Термін здачі»:

за замовчуванням встановлено на сьогоднішню дату плюс 7 днів.

г) кнопка «Файл»:

– дозволяє завантажити файл зі свого пристрою;

– може залишитись пустим.

ґ) кнопка «Зберегти»:

дозволяє зберегти інформацію.

6) створення відповіді:

а) поле «Відповідь»:

– максимальна довжина поля: 500 символів;

– може залишитись пустим.

б) кнопка «Файл»:

– дозволяє завантажити файл зі свого пристрою;

– може залишитись пустим.

в) кнопка «Відправити»:

дозволяє зберегти інформацію.

7) сторінка відгуку:

а) поле «Оцінка»:

– значення повинно бути в діапазоні від 1 до 5;

– обов’язкове для заповнення під час створення відгуку.

б) поле «Коментар»:

– максимальна довжина поля: 1000 символів;

– обов’язкове для заповнення під час створення відгуку.

в) кнопка «Відправити»:

дозволяє зберегти інформацію.

8) сторінка запису до репетитора:

а) випадаючий список «Предмет»:

обов’язкове для вибору під час створення запису.

б) поле «Початок»:

– обов’язкове для заповнення під час створення відгуку.

– час не може бути обраний у минулому

– має формат дати

в) поле «Кінець»:

– обов’язкове для заповнення під час створення відгуку.

– час не може бути обраний у минулому

– має формат дати

г) поле «Коментар»:

– максимальна довжина поля: 1000 символів;

– обов’язкове для заповнення під час створення відгуку.

ґ) кнопка «Підтвердити»:

дозволяє зберегти інформацію.

д) кнопка «Скасувати»:

дозволяє відмінити вибір часу.

9) головна сторінка пошуку репетиторів:

а) поле “Пошук”:

приймає символи, числа та знаки.

б) поля «Ціна за годину»:

– приймають значення від мінімуму до максимуму із запропонованих;

– може залишитись пустим.

в) розкриваючий список «Предмети»:

– список повинен містити всі існуючі категорії;

– має бути можливість обрати тільки одну категорію.

г) чек-бокси «Тип заняття»:

мають бути деактивовані за замовчуванням.

ґ) поле «Картка репетитора»

має надавати можливість переходити до профілю репетитора.

4.1.3 Для адміністратора системи (якщо він передбачений):

Адміністратори мають можливість створювати, видаляти та редагувати предмети та міста, а також видаляти профілі користувачів.

4.1.4 Додаткові вимоги:

– розробити програмне забезпечення за типом веб-застосунку;
– серверна частина має бути реалізована на мові програмування C#;
– використати в якості бази даних MySQL;
– для клієнтської частини використати JavaScript.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних. Вебсатосунок повинен бути перевірений на помилки і баги для забезпечення надійної роботи та уникнення негативних наслідків для користувачів.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.4 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.5 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.6 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Для користування вебзастосунком потрібно мати пристрій, на якому встановлено один із популярних браузерів.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

Рекомендована конфігурація технічних засобів:

– тип процесору: AMD Ryzen 5 3500U;
– об‘єм ОЗП: 16 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.7 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.) або Unix.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: HTTP запити з можливими URL параметрами та JSON значеннями у тілі запиту.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: відповіді на HTTP запити у форматі JSON та HTML.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування C# з використанням ASP.NET Core. Для веб-інтерфейсу використати JavaScript з використання Bootstrap, HTML і CSS для створення зовнішнього вигляду та стилізації користувацького інтерфейсу.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Visual Studio 2017.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді посилання на гітхаб.

4.5.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.5.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.5.8 Спеціальні вимоги

Не висуваються.

ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема бази даних;
– креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	11.03	
2.	Розробка технічного завдання	26.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	02.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	10.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	17.04	Тексти програмного забезпечення
6.	Обґрунтування вибору використаних технічних засобів	20.04	Технічна документація
7.	Розробка програмного забезпечення	11.05	Пояснювальна записка
8.	Налагодження програми	18.05	Технічна документація
9.	Розробка матеріалів графічної частини проекту	24.05	Графічний матеріал проекту

ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Вебзастосунок для пошуку репетиторів**

КПІ.ІТ-0221.045440.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	programming interface, прикладний програмний Інтерфейс
SDK	–	Software development kit
IT	–	Інформаційні технології
ER	–	Enity-Relation diagram
ОС	–	Операційна система.
СУБД	–	Система управління базами даних.

ВСТУП

Розвиток інформаційних технологій в останні десятиліття приніс значні зрушення в різні сфери життя, включаючи освіту. З появою Інтернету та веб-технологій зросла доступність інформації та можливість здійснювати різноманітні послуги онлайн. Одним із сегментів, який виявив потребу в інноваціях, є сфера репетиторських послуг.

Зважаючи на динаміку розвитку глобального ринку приватного репетиторства, яка описана в останньому звіті Міжнародної дослідницько-консалтингової групи з аналізу ринку [1] IMARC Group, очікуване зростання до 177,2 мільярда доларів США до 2028 року відображає стрімке підвищення попиту на репетиторські послуги.

Дослідження The Sutton Trust [2] демонструють широке поширення репетиторських послуг серед школярів у Великій Британії та Франції, що свідчить про важливість та актуальність розробки інструментів для пошуку кваліфікованих репетиторів. Також, зростання популярності онлайн-репетиторства в США відкриває нові можливості для розвитку вебзастосунків, спрямованих на забезпечення якісного навчання та розвитку особистості через інтерактивні та доступні платформи.

Ці аналітичні дані підкреслюють актуальність та значущість розробки вебзастосунку для пошуку репетиторів, який відповідає сучасним потребам освітнього сектору та сприяє підвищенню якості навчання. Враховуючи попит до репетиторства, метою є підвищення ефективності пошуку та підбору розкладу занять з репетитором, а запланований до розробки в даному проєкті вебзастосунок стане засобом досягнення цієї мети.

Вебсзастосунок був написаний мовою програмування C# [3] на платформі .NET з використанням ASP.NET Core. Для зберігання даних була використана СУБД MySQL [4], для роботи з даними було використано фреймворк Entity Framework Core [5].

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Робота зосереджена на створенні вебзастосунку, що спрямований на реалізацію пошуку репетиторів та організації навчального процесу. Вебзастосунки для пошуку репетиторів є актуальним напрямком у сучасній освітній сфері. На сьогоднішній момент існують певні недоліки у програмних рішеннях для пошуку репетиторів, зокрема, обмежена функціональність пошуку, нечіткість у критеріях відбору тощо.

У рамках дипломного проєкту було обрано напрямок розробки вебзастосунку для пошуку репетиторів з акцентом на розширення критеріїв пошуку, підбір розкладу занять з репетитором. Основна задача полягає в тому, щоб створити інструмент, який дозволить користувачам легко знаходити необхідних репетиторів, враховуючи різні параметри, такі як місце проведення занять, формат (онлайн або офлайн), вартість та предмет навчання. Крім того, особлива увага приділяється можливостям організації навчального процесу, включаючи виставлення завдань, запис на уроки та відслідковування виконання завдань.

Такий підхід дозволить покращити взаємодію між учнями та репетиторами, зробити процес навчання більш структурованим та продуктивним.

1.2 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації вебзастосунку для пошуку репетиторів. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.2.1 Аналіз відомих програмних продуктів

На сьогодні існує багато вебзастосунків і онлайн-платформ, які були розроблені для того, щоб люди могли легко знайти репетитора. Наступний аналіз

розгляне деякі з таких вебзастосунків і визначить їхні переваги та недоліки, які будуть виправлені в цій розробці.

1.2.1.1 Preply

Preply – це платформа з онлайн-навчання, де студенти можуть знаходити викладачів для вивчення різних предметів через відео-зв'язок. Функціонал даного програмного рішення включає в себе пошук та вибір викладачів з використанням фільтрації за предметом, ціною, країною народження та бажаним часом заняття, коментування та виставлення рейтингу викладачу, вибір бажаного часу для бронювання заняття тривалістю двох типів – 25 та 50 хвилин із обраним репетитором та інше.

ЇЇ перевагою є наявність відеозв’язку, проте платформа підтримує процес навчання лише у форматі зіздвонів, має високу вартість та фіксований інтервал занять для всіх викладачів. Загалом, Preply є корисною платформою для першого пошуку онлайн-викладача.

1.2.1.2 BUKI

BUKI є вебзастосунком для пошуку репетиторів, що дозволяє класифікувати їх за предметом, містом та типом проведення. Підтримує бронювання занять тривалістю півгодини.

Після оформлення заявки з учнем через вказаний номер телефону має зв’язатись викладач для уточнення деталей заняття та бажаного місця проведення заняття. Великим недоліком для репетиторів є те, що учні самостійно встановлюють вартість уроку під час оформлення заявки.

1.2.1.3 Асоціація репетиторів України

Асоціація репетиторів України – вебзастосунок для пошуку репетиторів з фільтрацією за містом, типом проведення заняття, статтю. Репетитора та мінімальною ціною, що дозволяє оформити заявку до обраного викладача із зазначенням бажаної кількості занять та їх тривалістю на тиждень. Однак вибору зручного часу занять із зазначених та пробних занять немає.

Для порівняння проєкту з аналогом можна скористатись таблицею 1.3.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	TutorSearch	Preply	BUKI	Асоціація репетиторів України	Пояснення
Реєстрація учня на платформі	+	+	-	-	Створення кабінету учня
Вибір зручного часу заняття	+	+	+	-	Можливість обрати бажаний час із зазначених
Виставлення завдань учням	+	-	-	-	Можливість для викладача залишати учням завдання та перевіряти їх виконання

Продовження таблиці 1.1

Функціонал	TutorSearch	Preply	BUKI	Асоціація репетиторів України	Пояснення
Бронювання офлайн-занять	+	-	+	+	Можливість підтримувати роботу викладачів офлайн

Найбільшим недоліком усіх існуючих аналогів є те, що для контролю навчання викладачам доводиться використовувати сторонні ресурси, де можливо виставляти завдання постійним студентам та передивлятись завантажені рішення.

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

При розробці вебзастосунку не використовувалося спеціальне алгоритмічне програмне забезпечення.

З точки зору архітектурних рішень, існує 2 підходи:

– мікросервісна архітектура – це архітектура, в якій проект розділений на ряд незалежних сервісів, кожна з яких виконує певні функції, що взаємодіють з іншими службами через певний інтерфейс. Перевага такої архітектури полягає в тому, що кожен мікросервіс може розвиватися самостійно та масштабуватися за необхідності, забезпечуючи більшу гнучкість у розробці та масштабуванні. Однак архітектура мікросервісів може ускладнити систему, особливо з точки зору мереж та управління послугами.

– монолітна архітектура передбачає створення єдиної системи, де всі компоненти програми тісно інтегровані та працюють в єдиному середовищі. Перевагами монолітної архітектури є простота розробки та розгортання, оскільки все включено в один пакет, що полегшує управління залежностями та координацію між різними частинами системи. Монолітні програми, легше розробляти та

підтримувати на ранніх стадіях проекту. Усі виклики між компонентами здійснюються локально в межах одного процесу, що усуває мережеві затримки і підвищує швидкодію системи. До того ж завдяки єдиній базі коду тестування монолітного додатка може бути простішим і з меншими витратами, оскільки не має необхідності розгортати та тестувати багато окремих служб.

Проаналізувавши обидві архітектури, було вирішено використовувати монолітну архітектуру.

Розроблене програмне забезпечення реалізує клієнт-серверну архітектуру (Client-Server). Клієнт-серверна архітектура - це розподілена обчислювальна система, в якій завдання розподіляються між програмним забезпеченням на сервері та клієнтським комп'ютером. Сервер розміщує, надає та керує більшістю ресурсів та послуг, що споживаються клієнтами. У системах, побудованих таким чином, може бути один або кілька клієнтських пристроїв, підключених до сервера через мережу. У зв’язку з тим, що всі послуги та запити надаються через мережу, такі системи вважаються розподіленими обчисленнями, де компоненти працюють незалежно один від одного.

Наявність гарної архітектури є ключем до створення програми. Завжди доступні різні типи архітектури, які мають однакові цілі, тобто розділення коду. Це можна зробити, розділивши додаток на шари.

У загальній архітектурі ASP.NET Core класифікуються два типи:

1) традиційна N-рівнева архітектура

Він має такі рівні, як UI, BLL (рівень бізнес-логіки) і DAL (рівень доступу до даних). Презентація будь-якої сторінки буде частиною рівня інтерфейсу користувача. На рівні інтерфейсу користувача користувачі можуть робити запити. BLL взаємодіє з цими запитами. Це залежить від DAL. BLL зберігає логіку в додатку. BLL викликає DAL для запитів доступу до даних. DAL забезпечує доступ до даних для всіх деталей реалізації. DAL залежить від існування бази даних. Одним із ключових недоліків цієї архітектури є те, що залежності часу компіляції проходять зверху вниз. Основний рівень інтерфейсу користувача залежить від BLL, а цей BLL залежить від DAL. Це означає, що BLL, який містить найважливішу

логіку в додатку, залежить від рівня доступу до даних. Таким чином, тестування бізнес-логіки набагато складніше, оскільки воно завжди вимагає тестової бази даних. Як рішення використовується принцип інверсії залежностей.

На рисунку 1.1 зображено архітектуру N-Layer.

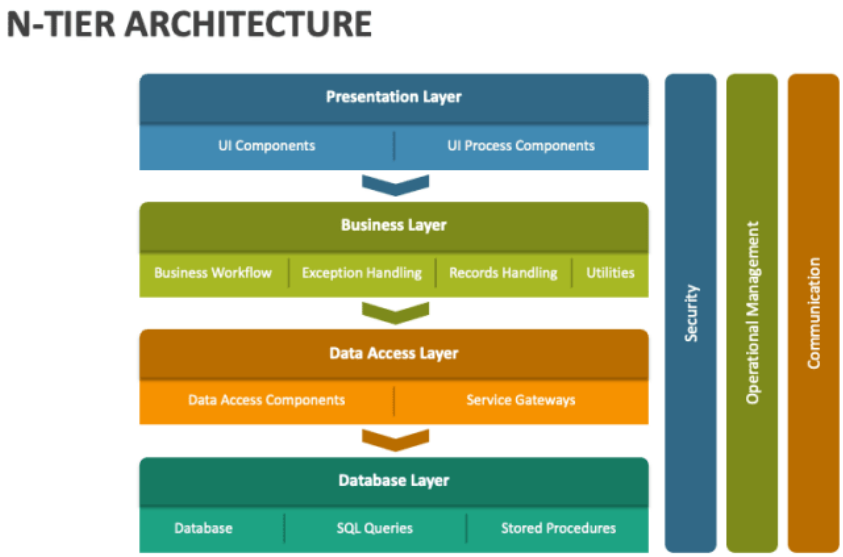


Рисунок 1.1 – Архітектура N-Layer

2) чиста архітектура (або цибулева)

У цьому типі доменний і прикладний рівні залишаються в основі дизайну, який називається центром дизайну. На відміну від традиційної архітектури «N-Layer», чиста архітектура не залежить від бази даних, рівня інтерфейсу користувача, фреймворку та кількох інших зовнішніх агентств. Чиста архітектура ставить бізнес-логіку та модель програми в центр програми. Він має обернену залежність, інфраструктура та деталі реалізації залежатимуть від ядра програми.[6]

Перевагами чистої архітектури можна вважати наступне:

а) інтерфейс користувача можна змінити в будь-який час, не змінюючи решту системи або бізнес-логіку.

б) бізнес-правила не прив’язані до будь-якої конкретної бази даних, тому для реалізації бізнес-правил можна використовувати BigTable, MongoDB, CouchDB або щось інше. Крім того, чиста архітектура не залежить

від існування деяких бібліотек, які мають особливості навантаженого програмного забезпечення.

в) чиста архітектура добре тестується, можна перевірити бізнес-правила без будь-яких інших зовнішніх елементів або навіть без інтерфейсу користувача, веб-сервера чи бази даних.

У чистій архітектурі є 3 рівні: рівень домену або рівень інфраструктури знаходиться в центрі й оточений рівнем ядра програми, а зовнішній рівень складається з інтерфейсів користувача.

На рисунку 1.2 залежності під час компіляції представлені суцільними стрілками, а залежності лише під час виконання — пунктирними стрілками. Стане дуже легко писати автоматичні модульні тести, і причина в тому, що ядро програми не залежить від інфраструктури. Програми, у яких проекти інтерфейсу користувача, інфраструктура та ядро додатків працюють як єдине ціле, називаються монолітними програмами.

Інтерфейси визначені в ядрі програми. Під час компіляції рівень інтерфейсу користувача працює з ними. Рівень інфраструктури визначатиме типи реалізації, а рівень інтерфейсу користувача не повинен знати про ці типи. Однак, коли програма виконується, вона потребує цих типів впровадження. Повинні бути присутні типи реалізації, які мають бути пов’язані з основними інтерфейсами програми. Крім того, за допомогою ін’єкції залежностей ми зможемо робити вищезгадані речі.

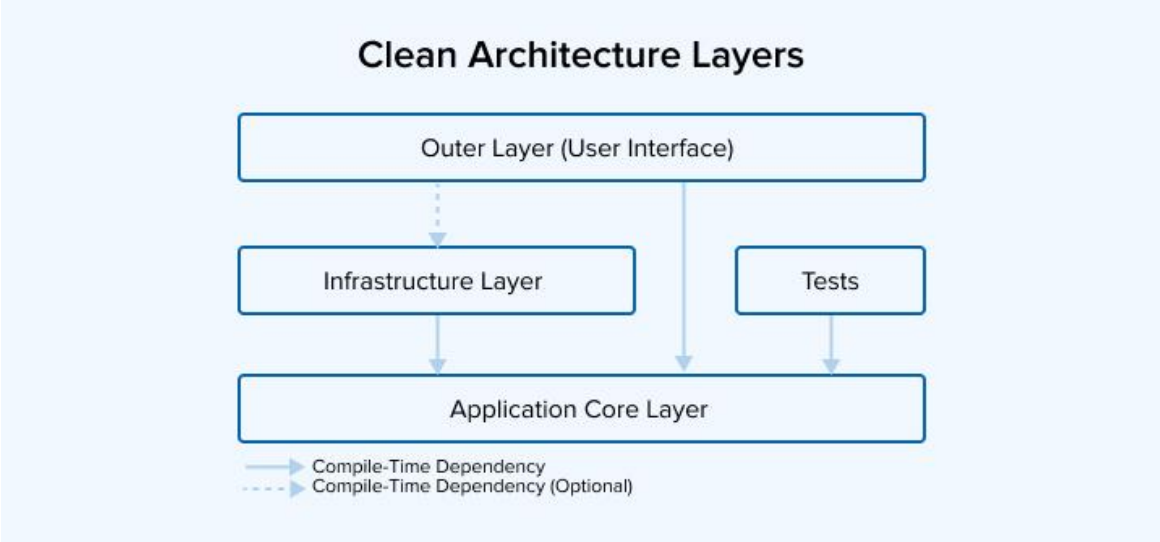


Рисунок 1.2 – Чиста архітектура

Для розробки було обрано патерн MVC (Model-View-Controller), що дозволяє розділити код на функціональні блоки, які відповідають за різні завдання. Ця модель дозволяє розмежувати три основні компоненти, що відповідають за бізнес-логіку, графічні інтерфейси та дані додатку. Перевагами шаблону проектування MVC є:

– розділення обов’язків: шаблон відокремлює інтерфейс користувача від логіки бізнесу та збереження даних;

– багатомовність: відокремлення бізнес-логіки від графічного інтерфейсу дозволяє підтримувати багатомовні додатки, що важливо для розширення цільової аудиторії;

– легка змінність: зміни в одному компоненті не впливають на інші компоненти, оскільки вони ізольовані, а отже, простіше робити модифікації;

– через поділ на компоненти спрощується етап тестування, також це дає можливість збільшення масштабу додатку заміною або дованням окремих складових, що свідчить про масштабованість.

На рисунку 1.3 зображено схему взаємодії рівнів патерну MVC.

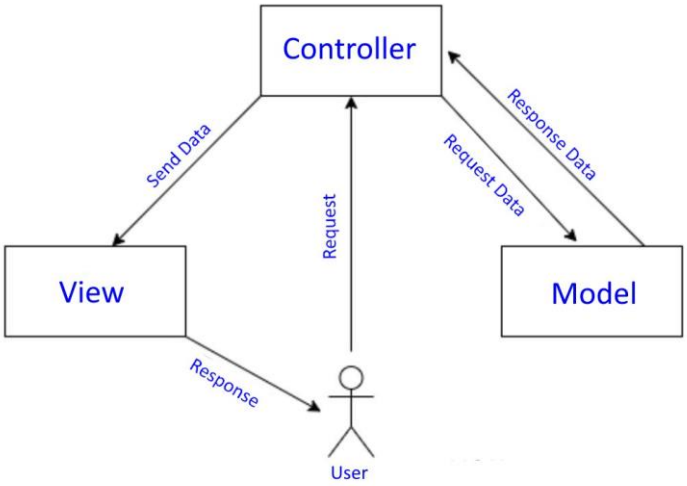


Рисунок 1.3 – Взаємодія рівнів шаблону проектування MVC

Також було використано патерн CQRS (Command and query responsibility segregation) – це принцип поділу читання і запису, який піднятий в модулях системи. У цьому принципі існують підсистеми, які займаються записом даних, і підсистеми, які займаються лише читанням даних. Ці підсистеми можуть використовувати абсолютно різні способи читання та запису даних, а в деяких випадках можуть використовуватися для різних баз даних.

Теоретично, така підсистема може використовувати тільки 1 базу даних. Якщо вам потрібно виконувати запити паралельно за принципом CQRS, то читання і запис використовуються для різних баз даних. При цьому події транслюються між цими базами даних. Як тільки в одній базі даних відбуваються зміни, ці зміни транслюються до всіх інших баз даних. В цьому випадку запис проводиться тільки з одного модуля, а зчитування може виконуватися паралельно з декількох модулів. Перевагами CQRS є:

– підвищення швидкості. Як правило, операція запису вимагає більше ресурсів, ніж операція читання. Розділивши модель на 2 частини, ви можете оптимізувати кожну операцію і знизити навантаження на систему.

– покращена масштабованість. Якщо ви розділите модель на 2 частини, ви зможете масштабувати кожну операцію незалежно від іншої. Наприклад, якщо

в системі є велика кількість запитів на читання, можна масштабувати операцію читання, не збільшуючи навантаження на операцію запису.

– більш точне моделювання бізнес-логіки. Поділ моделі на 2 частини дозволяє краще моделювати бізнес-логіку системи. Наприклад, операція читання може використовувати модель, яка більш точно відображає потреби користувача, тоді як операція запису може використовувати модель, яка більш точно відображає потреби бізнесу.

На рисунку 1.4 зображено схему роботи патерну CQRS.

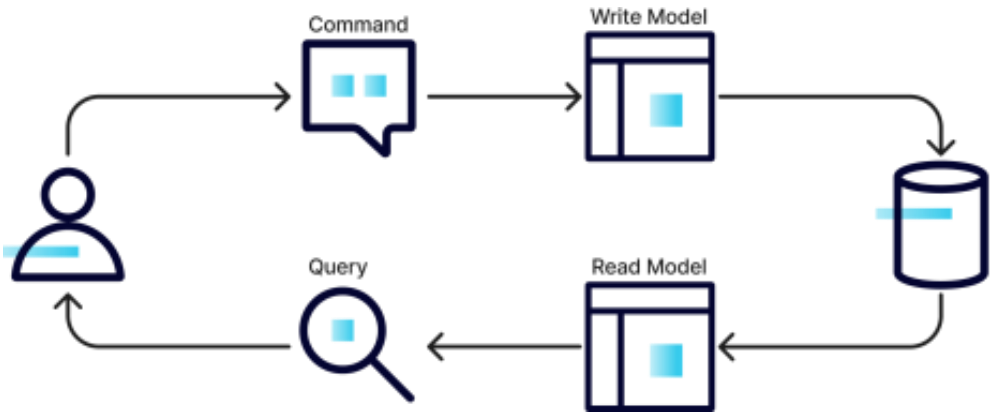


Рисунок 1.4 – Робота патерну CQRS

1.3 Опис бізнес-процесів

Для опису бізнес-процесу реєстрації нового користувача використовується BPMN модель, що відображена на рисунку 1.5.

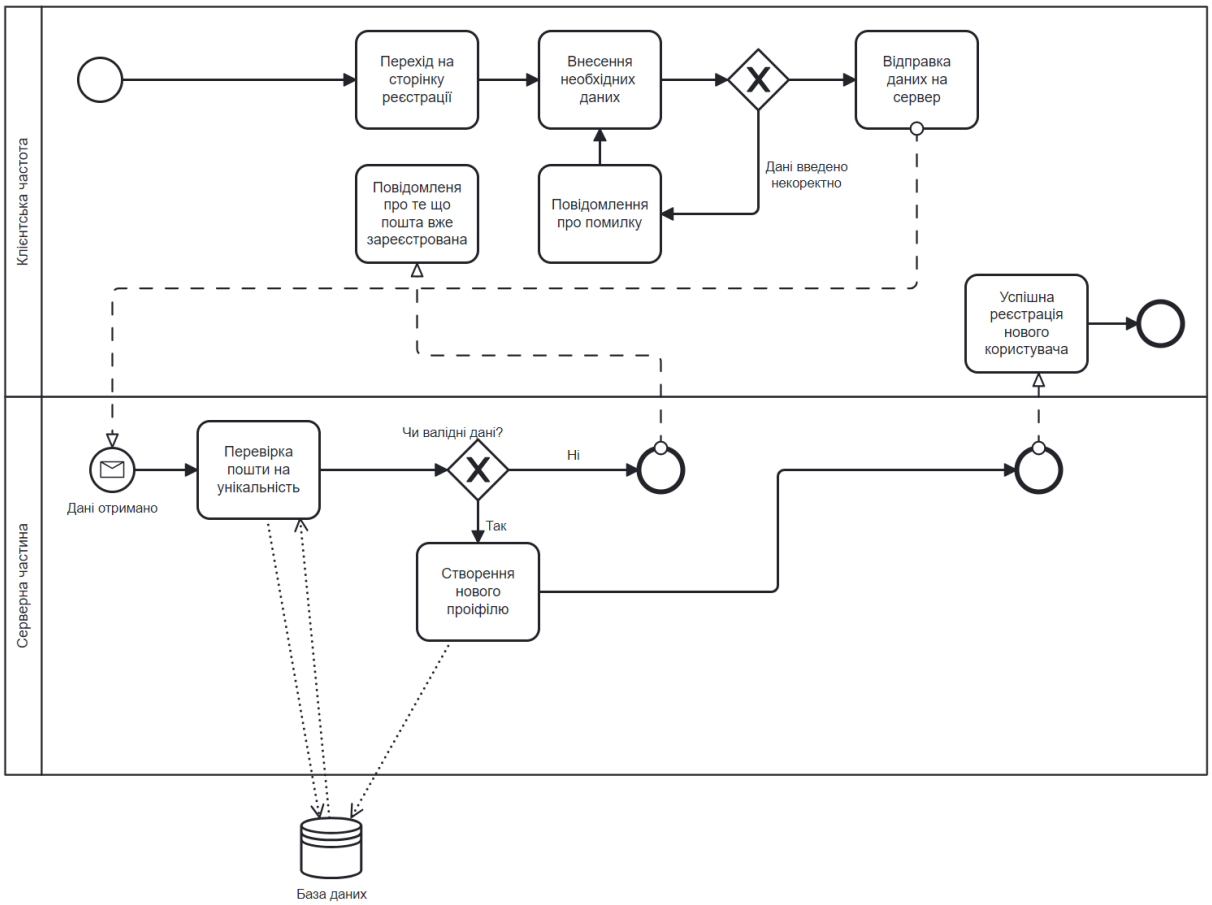


Рисунок 1.5 – Схема бізнес-процесу реєстрації користувача

Опис послідовності реєстрації нового користувача:

– користувач переходить на сторінку реєстрації;

– користувач заповнює дані для реєстрації: пошта, пароль та підтвердження паролю;

– запит обробляється сервісом, якщо введені поля, не відповідають шаблону заповнення, під відповідними полями з’явиться помилка;

– якщо введені дані відповідають шаблону заповнення клієнт отримує повідомлення про створення нового користувача.

Для опису бізнес-процесу авторизації користувача використовується BPMN модель, що відображена на рисунку 1.6.

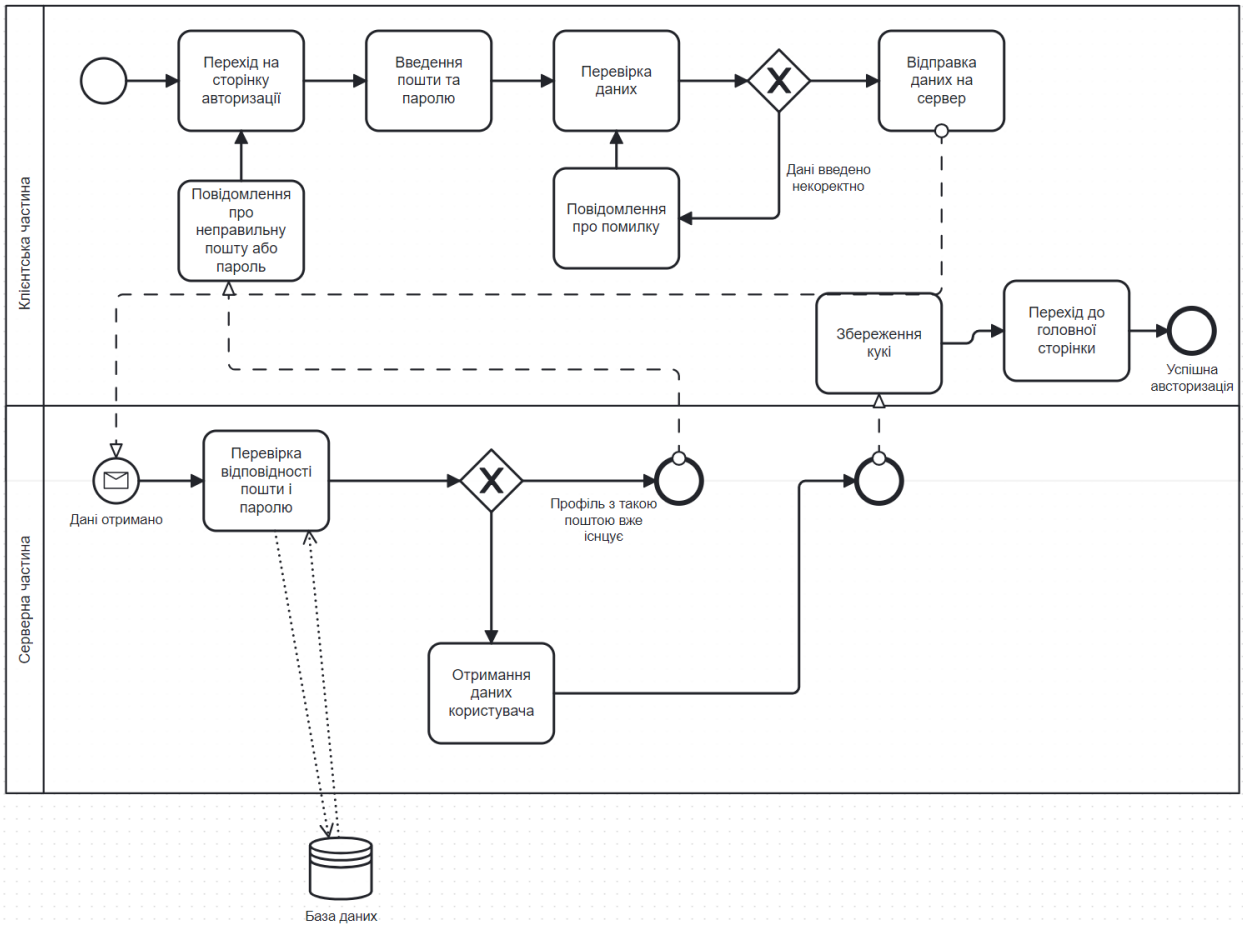


Рисунок 1.6 – Схема бізнес-процесу авторизації користувача

Опис послідовності асторизації користувача:

– користувач переходить на сторінку для входу;

– користувач вводить необхідні дані в поля реєстрації: пошту і пароль;

– запит обробляється сервісом, якщо введені дані не валідні, то сервіс повідомляє про помилку;

– на клієнтській частині, якщо від сервера прийшло повідомлення, що свідчить про помилку, то користувачу буде відображено цю помилку;

– якщо сервер повернув повідомлення, що свідчить про успіх, то клієнтська частина перенаправить користувача на головну сторінку.

Для опису бізнес-процесу пошуку репетиторів використовується BPMN модель, що відображена на рисунку 1.7.

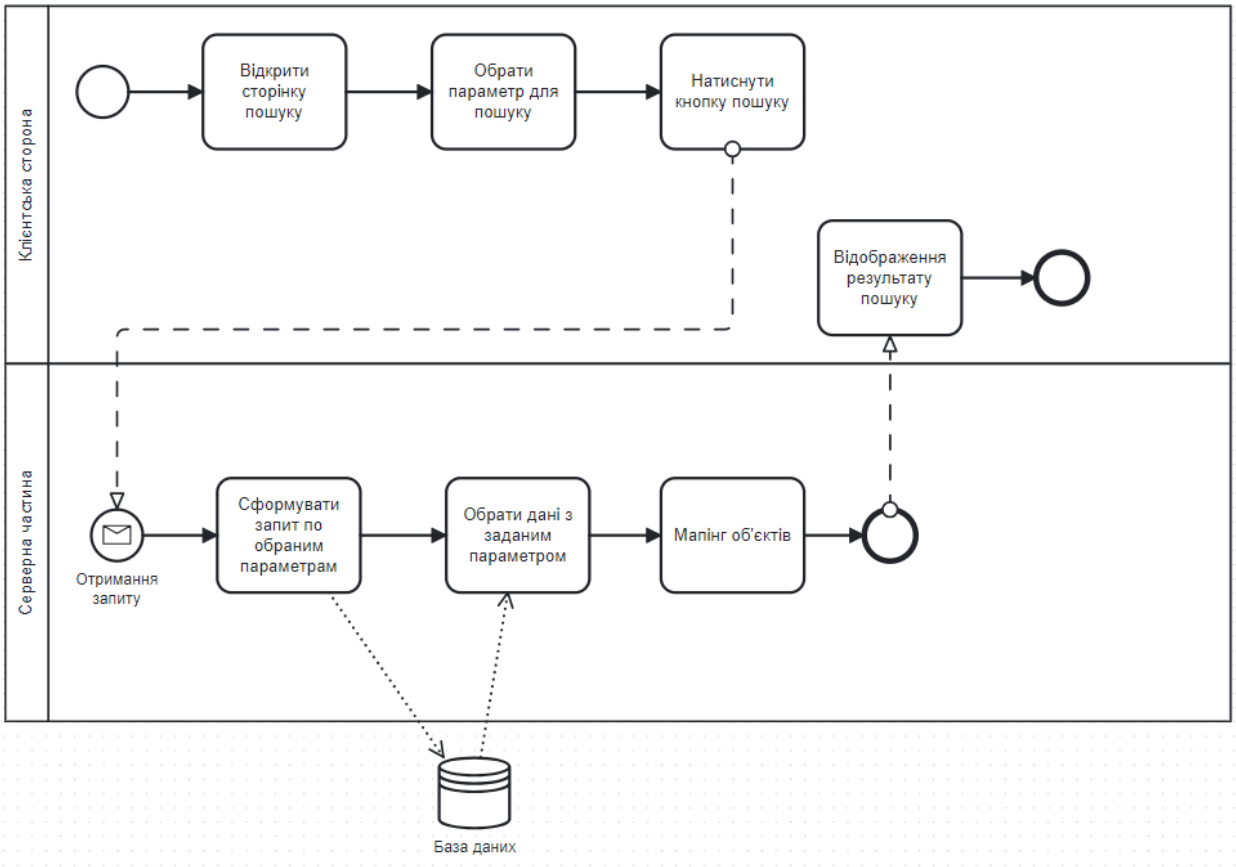


Рисунок 1.7 – Схема бізнес-процесу пошуку репетиторів

Опис послідовності пошуку репетиторів:

– користувач переходить на головну сторінку;

– користувач заповнює параметри для пошуку

– сервер отримує та формує запит до БД.

– відбувається мапінг об’єктів та список виводиться користувачу в користувацькому інтерфейсі.

Для опису бізнес-процесу створення запиту на урок використовується BPMN модель, що відображена на рисунку 1.8.

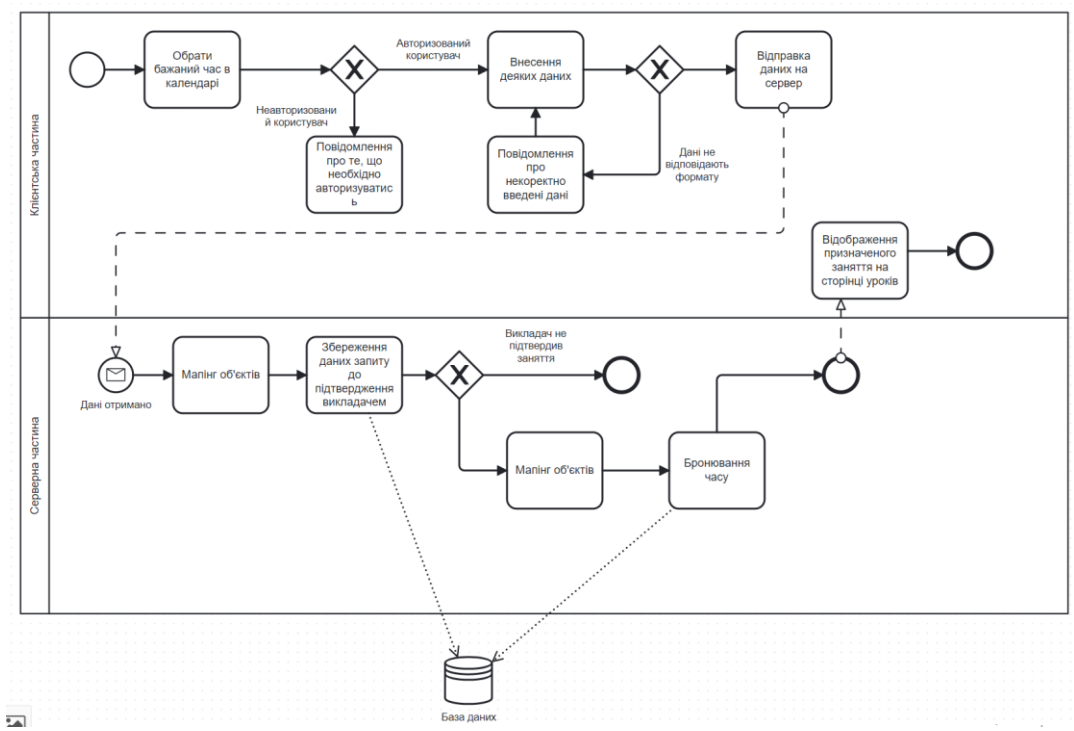


Рисунок 1.8 – Схема бізнес-процесу створення запиту на урок

Опис послідовності створення запиту на урок:

– користувач обирає години заняття в календарі;

– у випадку, якщо користувач не авторизований він отримає повідомлення помилки;

– користувач вносить дані для запису на урок;

– у випадку, якщо дані введено неправильно користувач отримає помилку;

– якщо дані валідні, створюється об’єкт, відбувається його мапінг;

– об’єкт зберігається в БД;

– після підтвердження заняття репетитором відбувається мапінг, об’єкт заняття зберігається в БД;

– користувацький інтерфейс відображає, що час заброньовано та урок планується на сторінці уроків.

Для опису бізнес-процесу створення завдання використовується BPMN модель, що відображена на рисунку 1.9.

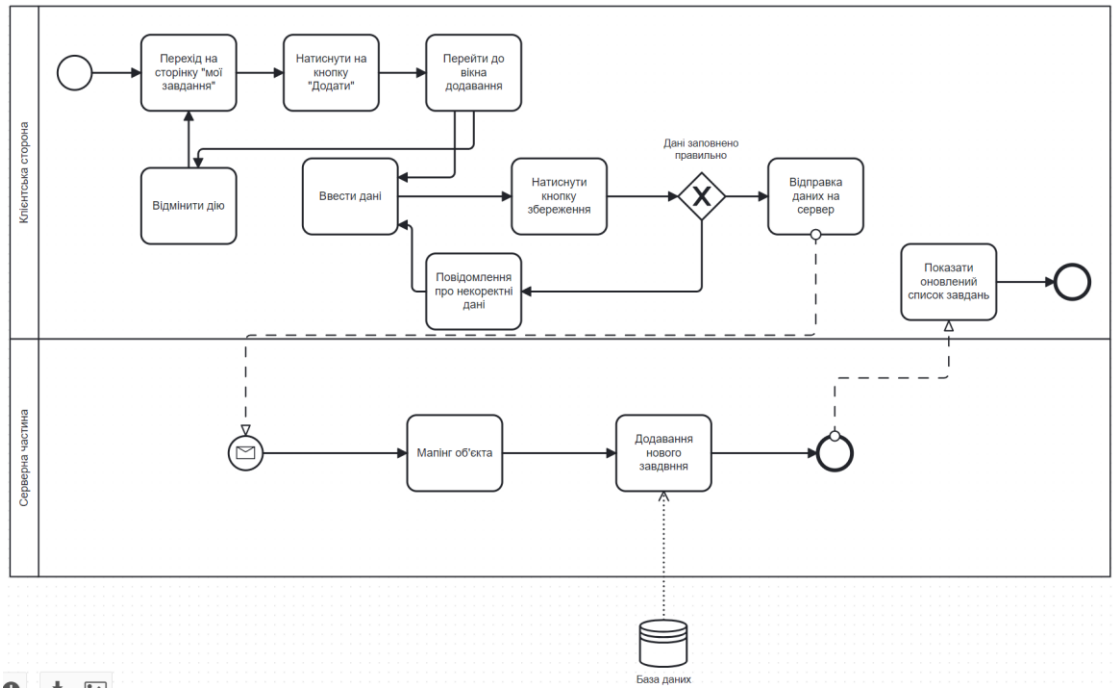


Рисунок 1.9 – Схема бізнес-процесу створення завдання

Опис послідовності створення завдання:

– користувач натискає кнопку створення завдання;

– при натисканні кнопки “Скасувати”, користувач повертається на сторінку своїх завдань;

– користувач заповнює дані для створення завдання та натискає кнопку “Зберегти”;

– при невдалій валідації даних користувач бачить відповідну помилку;

– якщо валідація проушла успішно, створюється об’єкт, відбувається його мапінг;

– об’єкт зберігається в БД;

– користувач бачить оновлений список завдань разом із створеним.

Для опису бізнес-процесу видалення завдання використовується BPMN модель, що відображена на рисунку 1.10.

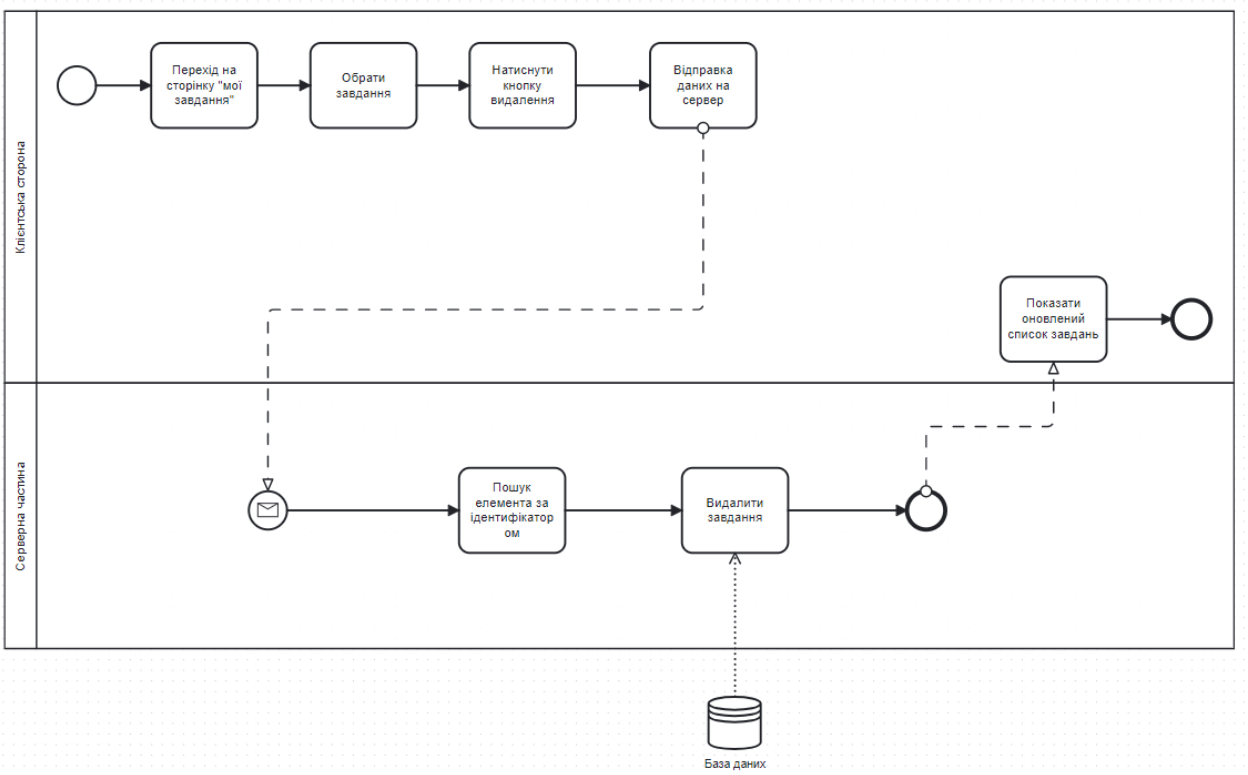


Рисунок 1.10 – Схема бізнес-процесу видалення завдання

Опис послідовності видалення завдання:

– користувач обирає завдання;

– користувач натискає кнопку видалення завдання;

– сервіс знаходить обране завдання по ідентифікатору та видаляє його з БД;

– користувач бачить оновлений список завдань без видаленого.

Для опису бізнес-процесу створення відповіді використовується BPMN модель, що відображена на рисунку 1.11.

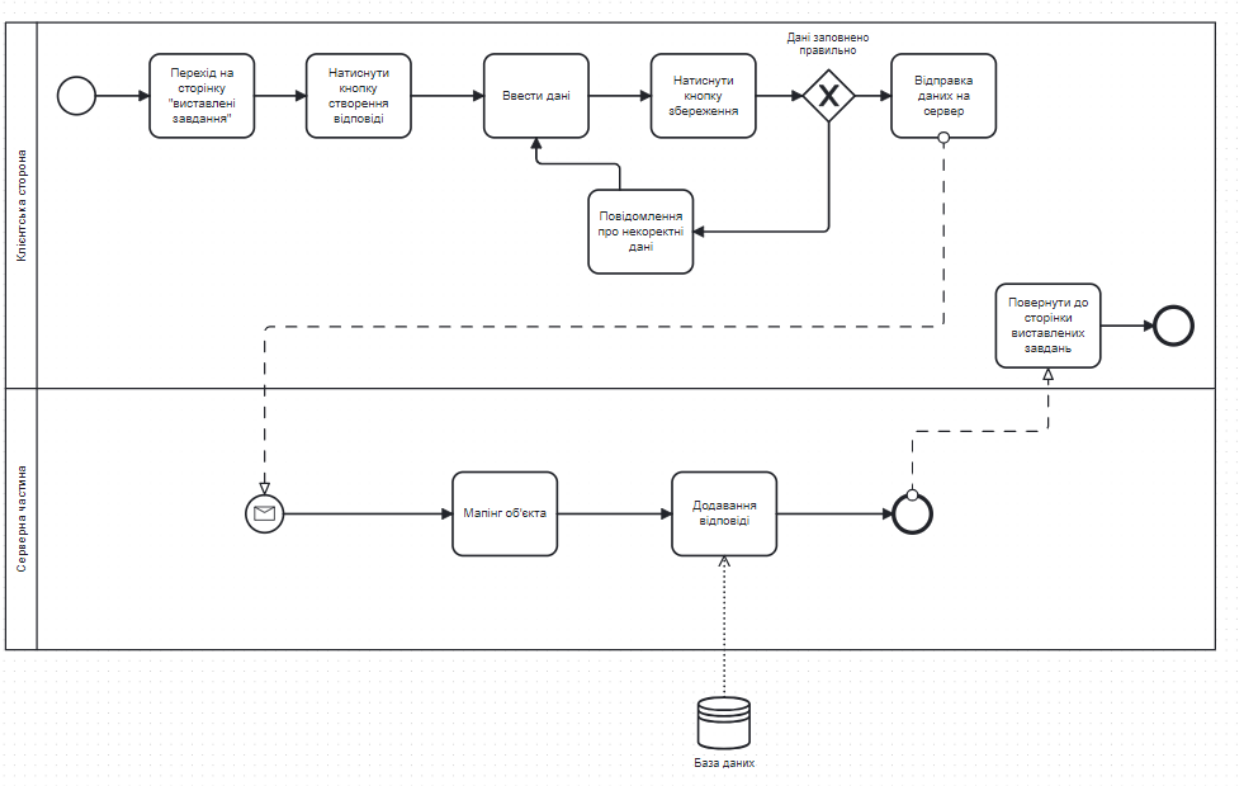


Рисунок 1.11 – Схема бізнес-процесу створення відповіді

Опис послідовності створення завдання:

– користувач натискає кнопку створення відповіді;

– при натисканні кнопки “Скасувати”, користувач повертається на сторінку виставлених завдань;

– користувач заповнює дані для створення відповіді та натискає кнопку “Зберегти”;

– при невдалій валідації даних користувач бачить відповідну помилку;

– якщо валідація пройшла успішно, створюється об’єкт, відбувається його мапінг;

– об’єкт зберігається в БД;

– користувач повертається до сторінки виставлених завдань, статус завдання змінюється відповідно на “На перевірці”.

1.4 Постановка задачі

Метою розробки є підвищення ефективності пошуку та підбору розкладу занять з репетитором. Такий інструмент значно полегшить процес пошуку відповідних спеціалістів, враховуючи індивідуальні потреби та вподобання користувачів. Однією з ключових особливостей вебзастосунку є можливість організації та управління навчальним процесом. Після проведення уроку репетитори зможуть виставляти завдання учням, що сприятиме закріпленню отриманих знань та розвитку навичок. Учні матимуть змогу прикріплювати відповіді до завдань безпосередньо у застосунку, що забезпечить зворотний зв’язок та можливість коригування навчального плану у режимі реального часу. Основні завдання, що підлягають розв’язанню в рамках розробки, включають:

– розробити інтерфейс для пошуку репетиторів та підбору розкладу занять;

– розробити функціонал для пошуку репетиторів. Вебзастосунок повинен реалізовувати фільтрацію за містом, типом заняття, вартістю, предметами;

– забезпечити сумісність з різними веб-браузерами та операційними системами, щоб користувачі могли використовувати його на будь-яких пристроях;

– розробити систему запису на уроки. Вебзастосунок повинен забезпечувати зручний спосіб планування розкладу занять, щоб уникати конфліктів у розкладі та забезпечити максимальну гнучкість для учнів і репетиторів;

– реалізувати підтримку зворотного зв’язку. Вебзастосунок повинен надавати можливість як учням, так і репетиторам залишати відгуки та оцінки, що сприятиме підвищенню якості освітніх послуг та допоможе новим користувачам зробити правильний вибір;

– розробити систему виставлення завдань та перегляду відповідей.

Висновки до розділу

У цьому розділі було проведено аналіз актуальності створення вебзастосунку для пошуку репетиторів, порівняно існуючі рішення з розробленим проектом та визначено основні бізнес-процеси та завдання.

Було виявлено, що найбільшим недоліком існуючих аналогів є те, що для контролю навчання викладачам доводиться використовувати сторонні ресурси, де можливо виставляти завдання постійним студентам та передивлятись рішення.

Для розгалуженої архітектури було обрано шаблони проектування, MVC та CQRS, що сприяє розділенню відповідальностей та підвищує гнучкість системи. Також було обрано чисту архітектуру через її переваги у розділенні відповідальностей, покращенні підтримки тестування, гнучкості та зручності підтримки коду. Вона дозволяє ефективно розділити бізнес-логіку, інтерфейс користувача та дані, що спрощує розробку та підтримку додатку, підвищуючи його стабільність та масштабованість.

Установлені основні завдання, які повинен вирішувати вебзастосунок: підвищення ефективності пошуку та вибору репетиторів, а також забезпечення зручного управління розкладом занять з ними.

Результати аналізу підтвердили необхідність та доцільність розробки вебзастосунку для пошуку репетиторів. Завдяки детальному аналізу та визначенню основних завдань, ми маємо чітке уявлення про шляхи подальшого розвитку цього проекту.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головними функціями програмного забезпечення є пошук репетиторів, створення свого акаунту репетитору, виставлення завдань для учнів, надання відповідей та їхній перегляд викладачем, інші функції відображено на рисунку 2.1.

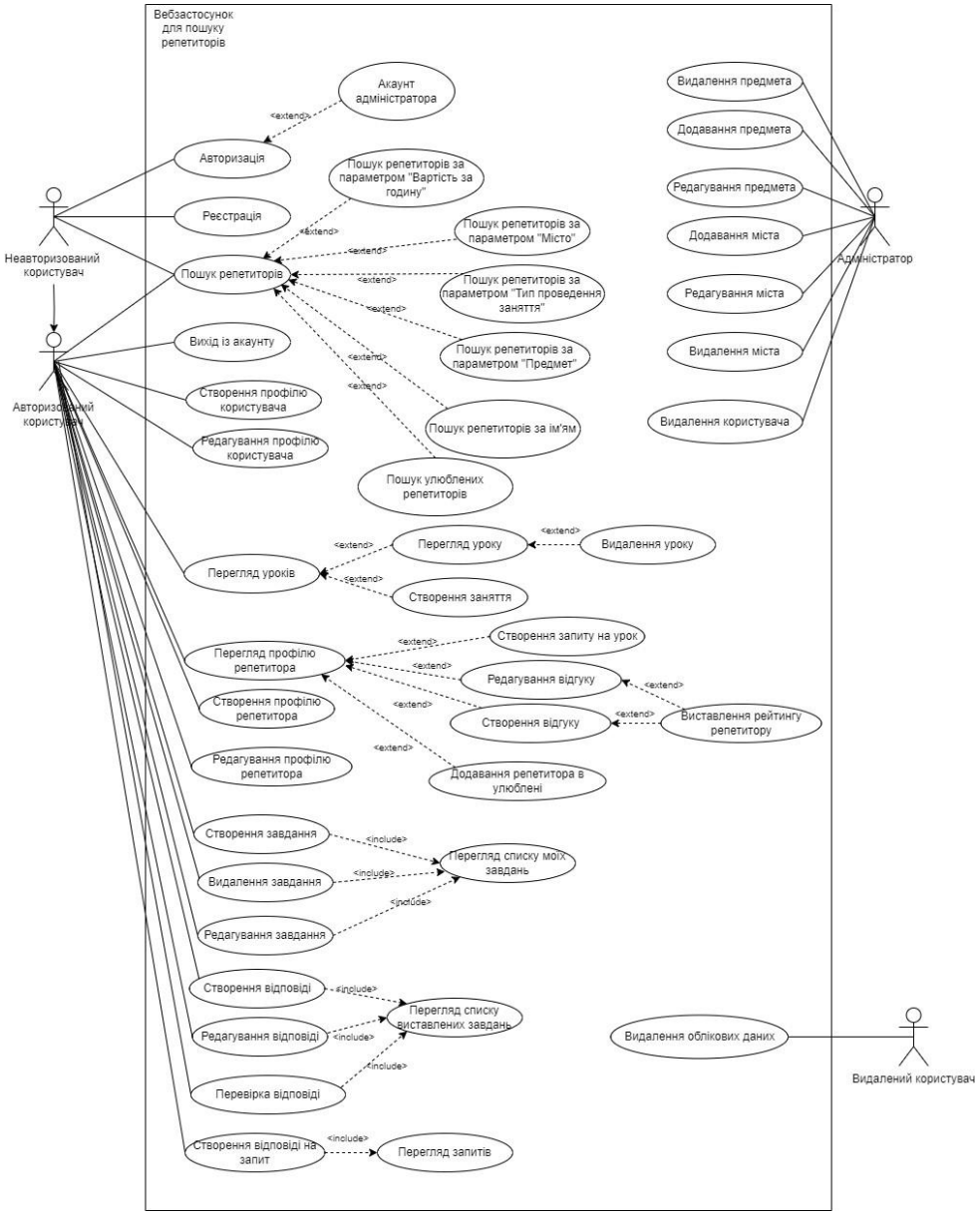


Рисунок 2.1 – Діаграма варіантів використання

У таблицях 2.1 – 2.37 наведені варіанти використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-1

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Створення нового облікового запису користувача
Actors	Незареєстрований користувач
Trigger	Користувач бажає зареєструватися
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку реєстрації. У поля для реєстрації вводяться наступні дані: пошта користувача, що ще ще не була використана у системі, пароль та підтвердження паролю, що відповідає паролю. Після заповнення полей сторінки реєстрації користувач натискає кнопку “Зареєструватися”. Після цього система перенапрявляє користувача на сторінку пошуку.
Extension	У випадку введення невалідних даних, відповідне поле підсвічується червоним написом та користувач побачить напис відповідної помилки.
Post-Condition	Створення облікового запису користувача, перехід на головну сторінку.

Таблиця 2.2 – Варіант використання UC-2

Use case name	Авторизація користувача
Use case ID	UC-02
Goals	Авторизація користувача за допомогою створеного раніше облікового запису
Actors	Неавторизований користувач
Trigger	Користувач бажає авторизуватися
Pre-conditions	Користувач вже має створений обліковий запис
Flow of Events	Користувач переходить на сторінку авторизації. Вводить свої пошту та пароль. Після чого користувач натискає кнопку “Авторизуватися”.
Extension	У випадку введення невалідних даних, відображення помилки “Недійсна спроба входу”.

Продовження таблиці 2.2

Post-Condition	Користувача авторизовано у вебзастосунку та перенаправлено на головну сторінку пошуку репетиторів.

Таблиця 2.3 – Варіант використання UC-3

Use case name	Вхід адміністратора
Use case ID	UC-03
Goals	Авторизація адміністратора у системі
Actors	Адміністратор
Trigger	Адміністратор бажає авторизуватися
Pre-conditions	Профіль адміністратора створений.
Flow of Events	Адміністратор переходить на сторінку авторизації. Вводить пошту, пароль адміністратора та натискає кнопку для авторизації.
Extension	У випадку введення невалідних даних, відображення помилки “Недійсна спроба входу.”
Post-Condition	Адміністратора авторизовано у вебзастосунку.

Таблиця 2.4 – Варіант використання UC-4

Use case name	Вихід із системи.
Use case ID	UC-04
Goals	Вихід із облікового запису
Actors	Авторизований користувач, Адміністратор
Trigger	Користувач бажає вийти
Pre-conditions	Користувач авторизований.
Flow of Events	Користувач обирає опцію виходу з вебзастосунку. Застосунок виконує деавторизацію користувача.
Extension	-
Post-Condition	Авторизований користувач стає неавторизованим користувачем вебзастосунку.

Таблиця 2.5 – Варіант використання UC-5

Use case name	Пошук репетиторів.
Use case ID	UC-05
Goals	Відобразити список репетиторів.
Actors	Неавторизований користувач, Авторизований користувач

Продовження таблиці 2.5

Trigger	Користувач знаходиться на головній сторінці
Pre-conditions	Проводиться фільтрація репетиторів згідно запиту користувача.
Flow of Events	Користувач обирає опції фільтрації та натискає кнопку пошуку. Система відображає репетиторів згідно запиту користувача.
Extension	Сторінка пошуку має включати фільтрацію за предметом, вартістю за годину, типом проведення заняття, містом та відображати улюблених.
Post-Condition	Список репетиторів відображено на сторінці пошуку.

Таблиця 2.6 – Варіант використання UC-6

Use case name	Створення профілю репетитора
Use case ID	UC-06
Goals	Створити профіль репетитора
Actors	Авторизований користувач
Trigger	Користувач бажає створити профіль репетитора
Pre-conditions	Користувач авторизований, має профіль користувача
Flow of Events	1) Користувач переходить на сторінку профілю 2) Користувач активує профіль репетитора 3) Користувач заповнює дані профілю репетитора, а саме предмети, тип проведення занять, вартість за годину, домашню адресу, короткий опис, інформацію про себе та календар. 4) Система створює профіль репетитора.
Extension	У випадку введення невалідних даних, відображення необхідної помилки.
Post-Condition	Профіль репетитора створено.

Таблиця 2.7 – Варіант використання UC-7

Use case name	Редагування профілю репетитора
Use case ID	UC-07
Goals	Редагувати профіль репетитора
Actors	Авторизований користувач
Trigger	Користувач бажає редагувати профіль репетитора
Pre-conditions	Користувач авторизований та має профіль репетитора

Продовження таблиці 2.7

Flow of Events	1) Користувач переходить на сторінку профілю 2) Користувач активує профіль репетитора 3) Користувач редагує дані профілю репетитора, а саме предмети, тип проведення занять, вартість за годину, домашню адресу, короткий опис, інформацію про себе та календар за бажанням. 4) Оновлення данихпрофілю у базі даних.
Extension	У випадку введення невалідних даних, відображення необхідної помилки.
Post-Condition	Профіль репетитора редаговано та дані оновлено.

Таблиця 2.8 – Варіант використання UC-8

Use case name	Перевірка відповіді учня
Use case ID	UC-08
Goals	Перевірити відповідь учня
Actors	Авторизований користувач
Trigger	Користувач бажає перевірити відповідь учня
Pre-conditions	Користувач авторизований, має профіль репетитора та отримав відповідь на виставлене завдання від учня.
Flow of Events	1) Користувач переходить на сторінку завдань, обирає необхідне із статусом “На перевірці” та натискає на кнопку перегляду. 2) Після перегляду репетитор залишає коментар до виконаної роботи та обирає значення статусу із випадаючого списку відповідно до роботи(“Виконано”, “На перевірці”, “До виконання”). 3) Система оновлює відповідь.
Extension	У випадку введення невалідниxx даних, відображення необхідної помилки.
Post-Condition	Відповідь перевірено та прокоментовано.

Таблиця 2.9 – Варіант використання UC-9

Use case name	Перегляд профілю репетитора.
Use case ID	UC-09
Goals	Переглянути профіль обраного репетитора
Actors	Неавторизований користувач, Авторизований користувач

Продовження таблиці 2.9

Trigger	Користувач бажає переглянути профіль обраного репетитора.
Pre-conditions	Користувач знаходиться на головній сторінці вебзастосунку.
Flow of Events	Користувач натискає на обраного репетитора. Сервер шукає відповідний запис у базі даних та повертає його на клієнтську частину.
Extension	-
Post-Condition	Профіль обраного репетитора відображений користувачу

Таблиця 2.10 – Варіант використання UC-10

Use case name	Створення запиту на урок
Use case ID	UC-10
Goals	Створення запиту від користувача на урок репетитора
Actors	Авторизований користувач
Trigger	Користувач бажає створити заняття до репетитора
Pre-conditions	Користувач знаходиться у профілі обраного репетитора.
Flow of Events	Користувач обирає зручний час заняття з доступного. Сервер зберігає дані в базі даних.
Extension	У випадку помилки на сервері користувачу буде відображена помилка.
Post-Condition	Заявка на урок відправлена репетитору.

Таблиця 2.11 – Варіант використання UC-11

Use case name	Створення відгуку
Use case ID	UC-11
Goals	Додати відгук про репетитора
Actors	Авторизований користувач
Trigger	Користувач бажає залишити відгук про репетитора
Pre-conditions	Користувач авторизований, знаходиться у профілі обраного репетитора та мав щонайменше одне заняття із обраним репетитором.
Flow of Events	1) Користувач натискає кнопку додавання відгуку 2) Користувач заповнює сторінку відгуку, обирає рейтинг та вводить коментар до відгуку 3) Система створює відгук про репетитора

Продовження таблиці 2.11

Extension	У випадку відсутності занять із обраним репетитором, відображення помилки “У вас не було жодного уроку з цим викладачем”. Відображення помилки у випадку спроби створення відгуку самому собі.
Post-Condition	Відгук створено.

Таблиця 2.12 – Варіант використання UC-12

Use case name	Редагування відгуку
Use case ID	UC-12
Goals	Редагувати відгук про репетитора
Actors	Авторизований користувач
Trigger	Користувач бажає змінити відгук про репетитора
Pre-conditions	Користувач авторизований, знаходиться у профілі обраного репетитора та мав щонайменше одне заняття із обраним репетитором.
Flow of Events	1) Користувач натискає кнопку додавання відгуку 2) Користувач змінює сторінку відгуку, обирає інший рейтинг та коментар до відгуку за бажанням 3) Система змінює відгук про репетитора
Extension	У випадку помилки на сервері користувачу буде відображена помилка.
Post-Condition	Відгук змінено.

Таблиця 2.13 – Варіант використання UC-13

Use case name	Створення профілю користувача
Use case ID	UC-13
Goals	Створити профіль користувача
Actors	Авторизований користувач
Trigger	Користувач бажає створити профіль користувача
Pre-conditions	Користувач авторизований
Flow of Events	1) Користувач переходить на сторінку профілю 2) Користувач заповнює дані профілю користувача, а саме ім’я, прізвище, по-батькові та завантажує аватар за бажанням. 3) Система створює профіль користувача.

Продовження таблиці 2.13

Extension	У випадку введення невалідних даних, відображення відповідної помилки.
Post-Condition	Профіль користувача створено.

Таблиця 2.14 – Варіант використання UC-14

Use case name	Редагування профілю користувача
Use case ID	UC-14
Goals	Редагувати профіль користувача
Actors	Авторизований користувач
Trigger	Користувач бажає редагувати профіль користувача
Pre-conditions	Користувач авторизований та має профіль користувача
Flow of Events	1) Користувач переходить на сторінку профілю 2) Користувач редагує дані профілю користувача, а саме ім’я, прізвище, по-батькові та завантажує аватар за бажанням. 3) Сервер оновлює дані профілю у базі даних.
Extension	У випадку введення невалідних даних, відображення відповідної помилки.
Post-Condition	Профіль користувача редаговано та дані оновлено.

Таблиця 2.15 – Варіант використання UC-15

Use case name	Створення завдання
Use case ID	UC-15
Goals	Створити завдання для учня
Actors	Авторизований користувач
Trigger	Користувач бажає створити завдання для учня
Pre-conditions	Користувач авторизований та має профіль репетитора. У репетитора з учнем відбувся урок.
Flow of Events	1) Користувач переходить на сторінку своїх завдань 2) Користувач натискає кнопку створення завдання 3) Користувач заповнює поля для створення завдання, а саме назву завдання, опис завдання та прикріплює файл за потреби. 4) Система створює завдання.
Extension	У випадку введення невалідних даних, відображення відповідної помилки.

Продовження таблиці 2.15

Post-Condition	Завдання створено.

Таблиця 2.16 – Варіант використання UC-16

Use case name	Редагування завдання
Use case ID	UC-16
Goals	Редагувати завдання для учня
Actors	Авторизований користувач
Trigger	Користувач бажає редагувати завдання для учня
Pre-conditions	Користувач авторизований та має профіль репетитора. Репетитор має хоча б одне своє завдання.
Flow of Events	1) Користувач переходить на сторінку своїх завдань 2) Користувач обирає відповідне завдання 3) Користувач змінює дані полів для створення завдання, а саме назву завдання, опис завдання та прикриплює новий файл за бажанням. 4) Сервер оновлює дані у базі даних.
Extension	У випадку введення невалідних даних, відображення відповідної помилки.
Post-Condition	Завдання редаговано.

Таблиця 2.17 – Варіант використання UC-17

Use case name	Видалення завдання
Use case ID	UC-17
Goals	Видалення завдання для учня
Actors	Авторизований користувач
Trigger	Користувач бажає видалити завдання для учня
Pre-conditions	Користувач авторизований та має профіль репетитора. Репетитор має хоча б одне своє завдання.
Flow of Events	1) Користувач переходить на сторінку своїх завдань 2) Користувач обирає відповідне завдання 3) Користувач натискає кнопку для видалення завдання 4) Сервер виконує видалення запису про завдання в базі даних.

Продовження таблиці 2.17

Extension	У випадку помилки на сервері користувачу буде відображена помилка.
Post-Condition	Обране завдання видалено.

Таблиця 2.18 – Варіант використання UC-18

Use case name	Створення відповіді
Use case ID	UC-18
Goals	Створити відповідь учня на завдання
Actors	Авторизований користувач
Trigger	Користувач бажає створити відповідь на завдання
Pre-conditions	Користувач авторизований. Користувач має виставлене завдання.
Flow of Events	1) Користувач переходить на сторінку виставлених завдань 2) Користувач натискає кнопку створення відповіді 3) Користувач заповнює поля для створення відповіді, а саме заповнив поле відповіді та прикріплює файл за бажанням. 4) Система створює відповідь.
Extension	У випадку введення невалідних даних, відображення відповідної помилки.
Post-Condition	Відповідь створено.

Таблиця 2.19 – Варіант використання UC-19

Use case name	Редагування відповіді
Use case ID	UC-19
Goals	Редагувати відповідь на завдання
Actors	Авторизований користувач
Trigger	Користувач бажає редагувати відповідь на завданння
Pre-conditions	Користувач авторизований. Користувач має відповідь на хоча б одне завдання.
Flow of Events	1) Користувач переходить на сторінку виставлених завдань 2) Користувач натискає на кнопку відповіді 3) Користувач змінює дані поля відповіді та прикріплює новий файл за бажанням. 4) Сервер оновлює дані у базі даних.

Продовження таблиці 2.19

Extension	У випадку введення невалідних даних, відображення відповідної помилки.
Post-Condition	Відповідь редаговано.

Таблиця 2.20 – Варіант використання UC-20

Use case name	Перегляд списку уроків.
Use case ID	UC-20
Goals	Переглянути список уроків користувача.
Actors	Авторизований користувач
Trigger	Користувач бажає переглянути список уроків.
Pre-conditions	Користувач авторизований та має хоча б один схвалений урок.
Flow of Events	Користувач заходить на сторінку уроків. Сервер робить запит до бази даних, повертаючи список уроків користувача. Список відображається на клієнті.
Extension	
Post-Condition	Список уроків користувача відображено на сторінці уроків.

Таблиця 2.21 – Варіант використання UC-21

Use case name	Перегляд списків запитів.
Use case ID	UC-21
Goals	Переглянути список запитів користувача.
Actors	Авторизований користувач
Trigger	Користувач бажає переглянути список запитів.
Pre-conditions	Користувач авторизований та має хоча б один запит на заняття.
Flow of Events	Користувач заходить на сторінку запитів. Сервер робить запит до бази даних, повертаючи список запитів користувача та список запитів до користувача. Списки відображаються на клієнті.
Extension	-
Post-Condition	Список запитів користувача та до користувача відображено на сторінці запитів.

Таблиця 2.22 – Варіант використання UC-22

Use case name	Створення заняття
Use case ID	UC-22
Goals	Створити заняття для учня

Продовження таблиці 2.23

Actors	Авторизований користувач
Trigger	Користувач бажає створити заняття для учня
Pre-conditions	Користувач авторизований та має профіль репетитора.
Flow of Events	1) Користувач переходить на сторінку уроків 2) Користувач натискає кнопку створення заняття 3) Користувач заповнює поля для створення заняття, а саме дату, час початку та кінця заняття, коментар, обирає предмет та учня для заняття. 4) Система створює заняття.
Extension	У випадку введення невалідних даних, відображення відповідної помилки.
Post-Condition	Заняття створено.

Таблиця 2.23 – Варіант використання UC-23

Use case name	Перегляд списку виставлених завдань
Use case ID	UC-23
Goals	Переглянути список виставлених завдань.
Actors	Авторизований користувач
Trigger	Користувач бажає переглянути список виставлених завдань.
Pre-conditions	Користувач авторизований та має хоча б одне завдання.
Flow of Events	Користувач заходить на сторінку виставлених завдань. Сервер робить запит до бази даних, повертаючи список завдань. Списки відображаються на клієнті.
Extension	-
Post-Condition	Список завдань відображено на сторінці виставлених завдань.

Таблиця 2.24 – Варіант використання UC-24

Use case name	Перегляд списку своїх завдань
Use case ID	UC-24
Goals	Переглянути список своїх завдань.
Actors	Авторизований користувач
Trigger	Користувач бажає переглянути список своїх завдань.
Pre-conditions	Користувач авторизований та створив хоча б одне завдання учню.
Flow of Events	Користувач заходить на сторінку своїх завдань. Сервер робить запит до бази даних, повертаючи список створених користувачем завдань. Списки відображаються на клієнті.

Продовження таблиці 2.24

Extension	-
Post-Condition	Список завдань користувача відображено на сторінці своїх завдань.

Таблиця 2.25 – Варіант використання UC-25

Use case name	Додавання предмету
Use case ID	UC-25
Goals	Додати предмет викладання.
Actors	Адміністратор
Trigger	Адміністратор бажає додати предмет викладання
Pre-conditions	Адміністратор авторизований.
Flow of Events	1) Адміністратор переходить на сторінку списку предметів 2) Адміністратор натискає кнопку додавання предмету 3) Адміністратор вводить у поле назву предмету та натискає кнопку збереження. 4) Система оновлює список предметів.
Extension	-
Post-Condition	Предмет додано.

Таблиця 2.26 – Варіант використання UC-26

Use case name	Додавання міста
Use case ID	UC-26
Goals	Додати місто викладання.
Actors	Адміністратор
Trigger	Адміністратор бажає додати місто викладання
Pre-conditions	Адміністратор авторизований.
Flow of Events	1) Адміністратор переходить на сторінку списку міст 2) Адміністратор натискає кнопку додавання міста 3) Адміністратор вводить у поле назву міста та натискає кнопку збереження. 4) Система оновлює список міст.
Extension	-
Post-Condition	Місто додано.

Таблиця 2.27 – Варіант використання UC-27

Use case name	Редагування міста
Use case ID	UC-27
Goals	Редагувати місто викладання.
Actors	Адміністратор
Trigger	Адміністратор бажає редагувати місто викладання
Pre-conditions	Адміністратор авторизований.
Flow of Events	1) Адміністратор переходить на сторінку списку міст 2) Адміністратор натискає кнопку редагування міста 3) Адміністратор змінює назву міста та натискає кнопку збереження. 4) Система оновлює список міст.
Extension	-
Post-Condition	Місто редаговано.

Таблиця 2.28 – Варіант використання UC-28

Use case name	Редагування предмету
Use case ID	UC-28
Goals	Редагувати предмет викладання.
Actors	Адміністратор
Trigger	Адміністратор бажає редагувати предмет викладання
Pre-conditions	Адміністратор авторизований.
Flow of Events	1) Адміністратор переходить на сторінку списку предметів 2) Адміністратор натискає кнопку редагування обраного предмету 3) Адміністратор змінює назву предмету та натискає кнопку збереження. 4) Система оновлює список предметів.
Extension	-
Post-Condition	Предмет редаговано.

Таблиця 2.29 – Варіант використання UC-29

Use case name	Додавання репетитора в улюблені
Use case ID	UC-29
Goals	Додати репетитора в улюблені.

Продовження таблиці 2.29

Actors	Авторизований користувач
Trigger	Користувач бажає додати репетитора в улюблені
Pre-conditions	Користувач авторизований.
Flow of Events	1) Користувач переходить на сторінку профілю обраного репетитора 2) Користувач натискає кнопку додавання в улюблені репетитори 3) Система оновлює список улюблених репетиторів.
Extension	-
Post-Condition	Репетитора додано в улюблені.

Таблиця 2.30 – Варіант використання UC-30

Use case name	Додавання рейтингу репетитора
Use case ID	UC-30
Goals	Виставити рейтинг репетитору.
Actors	Авторизований користувач
Trigger	Користувач бажає додати рейтинг репетитору
Pre-conditions	Користувач авторизований та мав хоча б одне заняття з репетитором.
Flow of Events	1) Користувач переходить на сторінку профілю репетитора 2) Користувач натискає кнопку створення відгуку та обирає рейтинг від 1 до 5. 3) Система оновлює рейтинг репетитора як середній між виставленими учнями.
Extension	У випадку відсутності даних, відображення відповідної помилки.
Post-Condition	Рейтинг виставлено.

Таблиця 2.31 – Варіант використання UC-31

Use case name	Створення відповіді на запит
Use case ID	UC-31
Goals	Створити відповідь на запит до заняття.
Actors	Авторизований користувач
Trigger	Користувач бажає створити відповідь на запит уроку
Pre-conditions	Користувач авторизований. Користувач має викладати хоча б один предмет.

Продовження таблиці 2.31

Flow of Events	1) Користувач переходить на сторінку запитів 2) Користувач натискає кнопку відповіді на запит 3) Користувач заповнює поле коментаря та обирає статус запиту із випадаючого списку, а саме “Схвалено” чи “Відхилено” 4) Система оновлює запит.
Extension	У випадку введення невалідних даних, відображення відповідної помилки.
Post-Condition	Відповідь на запит створено.

Таблиця 2.32 – Варіант використання UC-32

Use case name	Перегляд уроку.
Use case ID	UC-32
Goals	Переглянути обраний урок зі списку уроків
Actors	Авторизований користувач
Trigger	Користувач бажає переглянути обраний урок.
Pre-conditions	Користувач знаходиться на сторінці уроків.
Flow of Events	Користувач натискає на обраний урок. Сервер шукає відповідний запис у базі даних та повертає його на клієнтську частину.
Extension	-
Post-Condition	Обраний урок відображено користувачу

Таблиця 2.33 – Варіант використання UC-33

Use case name	Видалення уроку
Use case ID	UC-33
Goals	Видалення уроку
Actors	Авторизований користувач
Trigger	Користувач бажає видалити урок
Pre-conditions	Користувач авторизований та має хоча б один урок.
Flow of Events	1) Користувач переходить на сторінку уроків 2) Користувач обирає відповідний урок 3) Користувач натискає кнопку для видалення уроку 4) Сервер виконує видалення запису про урок в базі даних.
Extension	У випадку помилки на сервері користувачу буде відображена помилка.
Post-Condition	Обраний урок видалено.

Таблиця 2.34 – Варіант використання UC-34

Use case name	Видалення предмету
Use case ID	UC-34
Goals	Видалити предмет викладання зі списку предметів.
Actors	Адміністратор
Trigger	Адміністратор бажає видалити предмет викладання
Pre-conditions	Адміністратор авторизований.
Flow of Events	1) Адміністратор переходить на сторінку обраного предмету 2) Адміністратор натискає кнопку видалення обраного предмету 3) Сервер виконує видалення запису про предмет в базі даних.
Extension	-
Post-Condition	Предмет видалено.

Таблиця 2.35 – Варіант використання UC-35

Use case name	Видалення міста
Use case ID	UC-35
Goals	Видалити місто викладання зі списку міст.
Actors	Адміністратор
Trigger	Адміністратор бажає видалити місто викладання
Pre-conditions	Адміністратор авторизований.
Flow of Events	1) Адміністратор переходить на сторінку обраного міста 2) Адміністратор натискає кнопку видалення обраного міста 3) Сервер виконує видалення запису про місто в базі даних.
Extension	-
Post-Condition	Місто видалено.

Таблиця 2.36 – Варіант використання UC-36

Use case name	Видалення користувача
Use case ID	UC-36
Goals	Видалити користувача зі списку користувачів.
Actors	Адміністратор
Trigger	Адміністратор бажає видалити користувача
Pre-conditions	Адміністратор авторизований.

Продовження таблиці 2.36

Flow of Events	1) Адміністратор переходить на сторінку списку користувачів 2) Адміністратор обирає користувача зі списку користувачів 3) Адміністратор натискає кнопку видалення 4) Сервер виконує видалення запису про користувача в базі даних.
Extension	-
Post-Condition	Користувача видалено.

Таблиця 2.37 – Варіант використання UC-37

Use case name	Видалення облікових даних
Use case ID	UC-37
Goals	Видалити користувача зі списку користувачів.
Actors	Видалений користувач
Trigger	Користувач бажає видалити облікові дані
Pre-conditions	Користувач авторизований та профіль користувача був видалений адміністратором
Flow of Events	1) Система відображає видаленому користувачу повідомлення про видалення профіля та пропонує видалення облікових даних 2) Користувач натискає кнопку видалення облікових даних 3) Сервер виконує видалення даних про користувача в базі даних.
Extension	-
Post-Condition	Облікові дані видалено.

2.2 Аналіз системних вимог

У таблиці 2.38 описані системні вимоги для стабільної роботи застосунку.

Таблиця 2.38 – Опис системних вимог

Назва вимоги	Рекомендовано
Процесор	AMD Ryzen 5 3500U;
Об’єм ОЗП	16 Гб;

Продовження таблиці 2.38

Швидкість підключення до мережі Інтернет	від 100 мегабіт;
Браузер	Google Chrome версією 89+, Firefox версією 87+, Safari версією 13+
Операційна система	Windows, Linux, MacOS

2.3 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.39 наведено загальну модель вимог, а в таблиці 2.40 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.2.

Таблиця 2.39 – Загальна модель вимог

№	Назва	ID вимоги	Пріоритети	Ризики
1	Система авторизації користувача.	FR-1	Високий	Високий
1.1	Реєстрація користувача.	FR-2	Високий	Високий
1.2	Авторизація користувача.	FR-3	Високий	Високий
1.3	Вихід із акаунту.	FR-4	Середній	Низький
1.4	Авторизація адміністратора.	FR-5	Середній	Середній
2	Система управління профілями репетиторів	FR-6	Високий	Високий
2.1	Пошук репетиторів за повним ім’ям репетитора	FR-7	Високий	Середній

Продовження таблиці 2.39

2.2	Фільтрація за предметом, містом, типом заняття та вартістю за годину.	FR-8	Середній	Середній
2.3	Перегляд списку улюблених репетиторів	FR-9	Середній	Низький
2.4	Створення профілю репетитора.	FR-10	Високий	Середній
2.5	Редагування профілю репетитора.	FR-11	Середній	Середній
2.6	Перегляд списку репетиторів.	FR-12	Високий	Низький
2.7	Перегляд профілю обраного репетитора.	FR-13	Високий	Низький
2.8	Створення відгуку про репетитора.	FR-14	Низький	Низький
2.9	Додавання рейтингу репетитору.	FR-15	Низький	Низький
2.10	Додавання репетитора в улюблені	FR-16	Низький	Низький
3	Система занять.	FR-17	Високий	Високий
3.1	Створення заняття.	FR-18	Високий	Середній
3.2	Перегляд списку занять.	FR-19	Високий	Низький
3.3	Створення відповіді на запит до заняття.	FR-20	Високий	Низький
3.4	Перегляд заняття.	FR-21	Середній	Низький
3.5	Видалення заняття.	FR-22	Високий	Низький
4	Система завдань.	FR-23	Високий	Високий
4.1	Створення завдання.	FR-24	Високий	Середній

Продовження таблиці 2.39

4.2	Видалення завдання.	FR-25	Середній	Низький
4.3	Редагування завдання.	FR-26	Середній	Середній
4.4	Перегляд списку завдань.	FR-27	Високий	Низький
4.5	Завантаження файлу до завдання.	FR-28	Середній	Високий
5	Система відповідей	FR-29	Високий	Високий
5.1	Створення відповіді	FR-30	Високий	Середній
5.2	Редагування відповіді	FR-31	Середній	Середній
5.3	Перевірка відповіді	FR-32	Середній	Низький
5.4	Завантаження файлу до відповіді.	FR-33	Середній	Високий
6	Система управління профілем користувача	FR-34	Середній	Середній
6.1	Створення профілю користувача	FR-35	Високий	Середній
6.2	Редагування профілю користувача	FR-36	Середній	Середній
7	Додавання предметів викладання	FR-37	Високий	Середній
8	Редагування предметів викладання	FR-38	Середній	Середній
9	Додавання міст викладання	FR-39	Високий	Середній
10	Редагування міст викладання	FR-40	Середній	Середній
11	Видалення предмета викладання	FR-41	Середній	Низький
12	Видалення міста викладання	FR-42	Середній	Низький
13	Видалення користувача	FR-43	Високий	Низький

Продовження таблиці 2.39

14	Видалення облікових даних	FR-44	Високий	Низький

Таблиця 2.40 – Перелік функціональних вимог

ID вимоги	Назва та опис
FR-1	Система авторизації користувача. Система має забезпечувати надійний механізм авторизації користувачів, який дозволяє ідентифікувати особу за допомогою унікальної пошти та пароля.
FR-2	Реєстрація користувача. Система має забезпечувати функціонал реєстрації нових користувачів за допомогою введення необхідних даних, а саме пошти, пароля та підтвердження пароля.
FR-3	Авторизація користувача. Система має забезпечувати функціонал авторизації користувачів, використовуючи дані вказані під час реєстрації.
FR-4	Вихід із акаунта. Система має забезпечувати можливість виходу авторизованих користувачів із будь-якої сторінки.
FR-5	Авторизація адміністратора. Система має забезпечувати функціонал авторизації адміністратора, використовуючи визначені облікові дані.
FR-6	Система управління профілями репетиторів. Система повинна дозволяти створювати свій профіль та взаємодіяти з профілями інших репетиторів.
FR-7	Пошук репетиторів за повним ім’ям репетитора. Система має дозволяти швидко знаходити профілі репетиторів заключовими словами в повному імені.

Продовження таблиці 2.40

FR-8	Фільтрація за предметом, містом, улюбленими репетиторами та вартістю за годину. Система має забезпечувати фільтрацію репетиторів за предметами викладання, вартістю за годину, містом викладання та типом викладання.
FR-9	Перегляд списку улюблених репетиторів. Система повинна забезпечити користувачам механізм перегляду улюблених репетиторів.
FR-10	Створення профілю репетитора. Система має забезпечити користувачам інтерфейс для створення свого профілю із вказанням необхідних даних, а саме предметів викладання, доступні години заняття, домашню адресу, короткий опис та інформацію про себе.
FR-11	Редагування профілю репетитора. Система має забезпечити користувачам інтерфейс для редагування даних свого профілю, а саме предметів викладання, доступні години заняття, домашню адресу, короткий опис та інформацію про себе.
FR-12	Перегляд списку репетиторів. Система повинна забезпечити користувачам механізм перегляду всіх доступних репетиторів.
FR-13	Перегляд профілю обраного репетитора. Користувачам має бути надана функція перегляду профілей репетиторів, що дозволяє переглядати особисту інформацію репетиторів.
FR-14	Створення відгуку про репетитора. Система повинна надавати можливість користувачам створити відгуки про репетиторів.

Продовження таблиці 2.40

FR-15	Додавання рейтингу репетитору. Система повинна надавати можливість користувачам додавати рейтинг про майданчики.
FR-16	Додавання репетитора в улюблені. Система повинна надавати можливість користувачам додавати репетиторів в улюблені.
FR-17	Система занять. Система повинна дозволяти створювати свої заняття та відхиляти чи приймати їх.
FR-18	Створення заняття. Система повинна надавати можливість користувачам створити запит на заняття, як зі сторони репетитора, так і зі сторони учня в доступний час.
FR-19	Перегляд списку занять. Система повинна забезпечити користувачам механізм перегляду всіх занять.
FR-20	Створення відповіді на запит до заняття. Система повинна надавати можливість користувачу додавати статус до запитів на свої заняття та залишати коментар.
FR-21	Перегляд заняття. Система повинна забезпечити користувачам механізм перегляду обраного заняття.
FR-22	Видалення заняття. Система повинна надавати можливість користувачам видаляти заняття, які більше не є актуальними або потрібними.
FR-23	Система завдань. Система повинна надавати можливість користувачам створити, видалити, редагувати та переглянути завдання.

Продовження таблиці 2.40

FR-24	Створення завдання. Система повинна надавати можливість користувачу із профілем репетитора створювати завдання учням, які мали хоча б одне заняття із ним.
FR-25	Видалення завдання. Система повинна надавати можливість користувачам видаляти завдання, які більше не є актуальними або потрібними.
FR-26	Редагування завдання. Система має дозволяти користувачам, що вже створили завдання раніше, модифікувати дані завдання.
FR-27	Перегляд списку завдань. Користувачі мають можливість переглядати повний список завдань, що були створені в системі.
FR-28	Завантаження файлу до завдання. Користувачі повинні мати змогу завантажувати файли до свого завдання.
FR-29	Система відповідей. Система повинна надавати можливість користувачам створити, редагувати та переглянути відповіді.
FR-30	Створення відповіді. Система повинна надавати можливість користувачу відповідь до виставленого їм завдання.
FR-31	Редагування відповіді. Система має дозволяти користувачам, що вже створили відповідь раніше, модифікувати дані відповіді.
FR-32	Перевірка відповіді. Користувачам має бути надана функція перевірки відповіді.

Продовження таблиці 2.40

FR-33	Завантаження файлу до відповіді. Користувачі повинні мати змогу завантажувати файли до своєї відповіді.
FR-34	Система управління профілем користувача. Система повинна надавати можливість користувачам створити, та редагувати профіль користувача.
FR-35	Створення профілю користувача. Система має реалізовувати функцію створення профілю користувача із заповненням полей імені, прізвища, по-батькові та можливістю завантажити аватар.
FR-36	Редагування профілю користувача Система має дозволяти користувачам, що вже створили профіль раніше, модифікувати дані профілю.
FR-37	Додавання предметів викладання. Система повинна надавати можливість адміністратору додавати предмети викладання.
FR-38	Редагування предметів викладання. Система повинна надавати можливість адміністратору редагувати предмети викладання.
FR-39	Додавання міст викладання. Система повинна надавати можливість адміністратору додавати міста викладання.
FR-40	Редагування міст викладання. Система повинна надавати можливість адміністратору редагувати міста викладання.
FR-41	Видалення предмету викладання. Система повинна надавати можливість адміністратору видаляти предмети викладання.

Продовження таблиці 2.40

FR-42	Видалення міста викладання. Система повинна надавати можливість адміністратору видаляти міста викладання.
FR-43	Видалення користувача. Система повинна надавати можливість адміністратору видаляти користувачів.
FR-44	Видалення облікових даних. Система повинна надавати можливість видаленому користувачу видаляти свої облікові дані.

Матрицю трасування вимог відображено на рисунку 2.2.

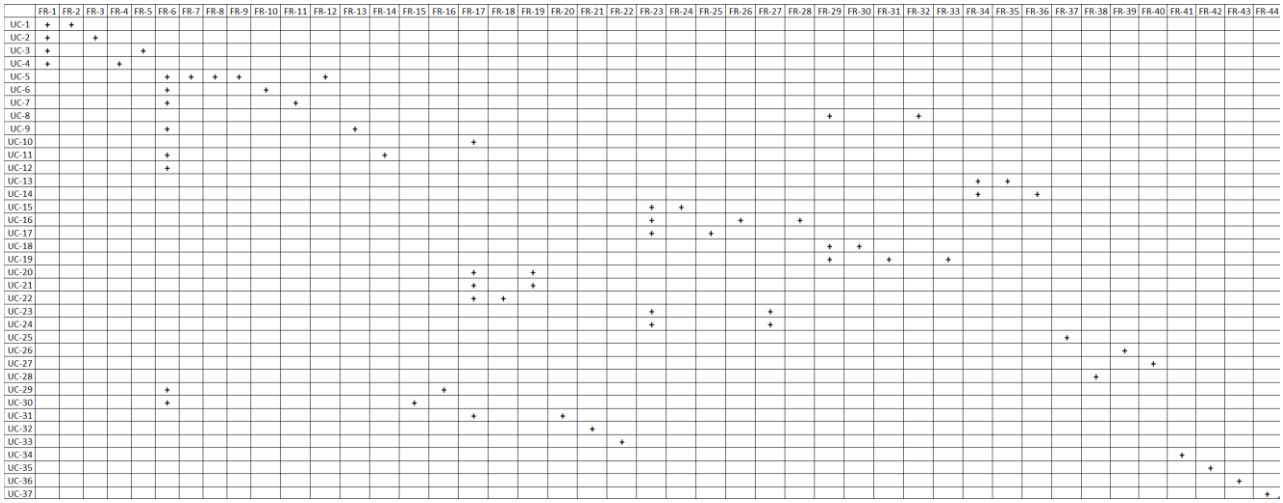


Рисунок 2.2 – Матриця трасування вимог

2.4 Розроблення нефункціональних вимог

Нефункціональні вимоги визначають обмеження та загальні характеристики системи, що мають вплив на її розробку та використання. Вони є ключовими для забезпечення якості та ефективності вебзастосунку.

Пріоритетними нефункціональними вимогами є:

– надійність. Система має реагувати та видавати коректний результат у випадку недоступності джерела даних, падіння процесу під створення звіту;

– легкість використання. Інтерфейс користувача має бути інтуїтивно зрозумілим та легким для навігації, щоб користувачі могли ефективно використовувати функціональність системи та легко взаємодіяти із нею;

– забезпечення безпеки даних користувачів. Система має впроваджувати відповідні механізми та протоколи для захисту персональних та інших чутливих даних користувачів, запобігаючи несанкціонованому доступу або витоку інформації.

Висновки до розділу

У ході даного розділу було проведено ретельний аналіз потреб користувачів та вимог до розроблюваної системи. Початкові вимоги були визначені, проаналізовані та деталізовані для подальшого розроблення програмного забезпечення.

Першим етапом проекту було визначення функціональних вимог, які включають в себе реєстрацію та авторизацію користувачів, управління профілями репетиторів, систему занять та завдань, а також систему відповідей та управління профілями користувачів. Ці вимоги були розбиті на модулі для більшої зручності та ефективності розробки програмного забезпечення.

Наступним кроком були визначені нефункціональні вимоги, серед яких найважливішими були надійність системи, легкість використання та забезпечення безпеки даних користувачів. Ці вимоги визначають обмеження та загальні характеристики системи, що мають вплив на її розробку та використання.

У результаті виконаної роботи була розроблена модель вимог, яка включає в себе детальний перелік функціональних та нефункціональних вимог до програмного забезпечення. Ця модель є основою для подальшого процесу розробки та впровадження системи, забезпечуючи відповідність її функціональності та характеристик потребам та очікуванням користувачів.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для вебзастосунку, що розробляється, обрано чисту архітектуру. Було прийнято таке рішення через мінімізацію залежностей між компонентами, гнучкість у зміні інтерфейсу користувача та легкість тестування.

Для відображення архітектури було обрано модель C4. Перевагою моделі C4 є розбиття архітектури на декілька рівнів, це робить модель простішою для читання та розуміння. Високорівнево архітектура вебзастосунку показана як контекстна діаграма на рисунку 3.1 та діаграма контейнерів на рисунку 3.2, що відповідає рівням 1 та 2 моделі C4 моделі представлення архітектури.

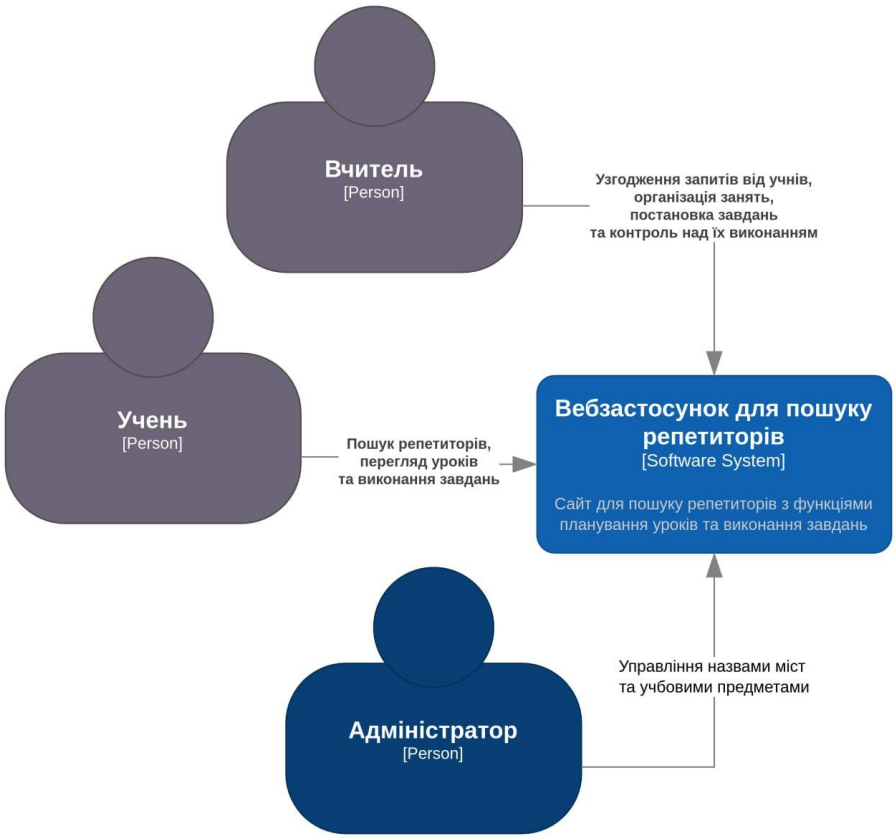


Рисунок 3.1 – Діаграма контексту

На першому рівні C4 моделі представлено діаграму контексту, на якій вказано взаємодію даної системи з користувачами застосунку.

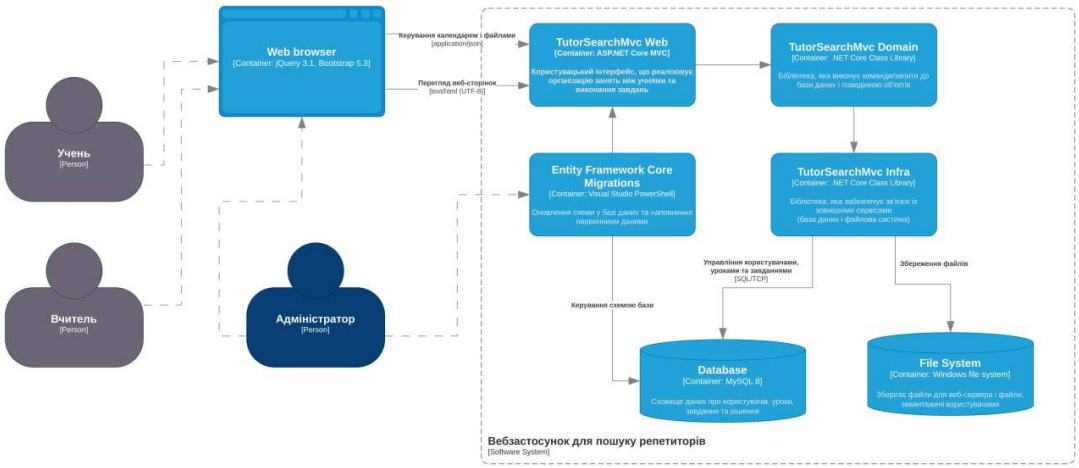


Рисунок 3.2 – Діаграма контейнерів

На другому рівні моделі C4 представлена діаграма контейнерів. Система складається з: контейнера бази даних, файлової системи, бібліотеки Infra, браузерного клієнта, бібліотеки Domain та механізму міграцій.

Для подальшої деталізації архітектури застосунку розділимо застосунок на три системи: система інфраструктурної бібліотеки, система доменної біблаотеки, система. Для глибшого розуміння кожної із систем виконаємо моделювання 3-го рівня C4 моделі для кожної із них. Взаємодію основних компонентів інфраструктурної бібліотеки зображено на рисунку 3.3.

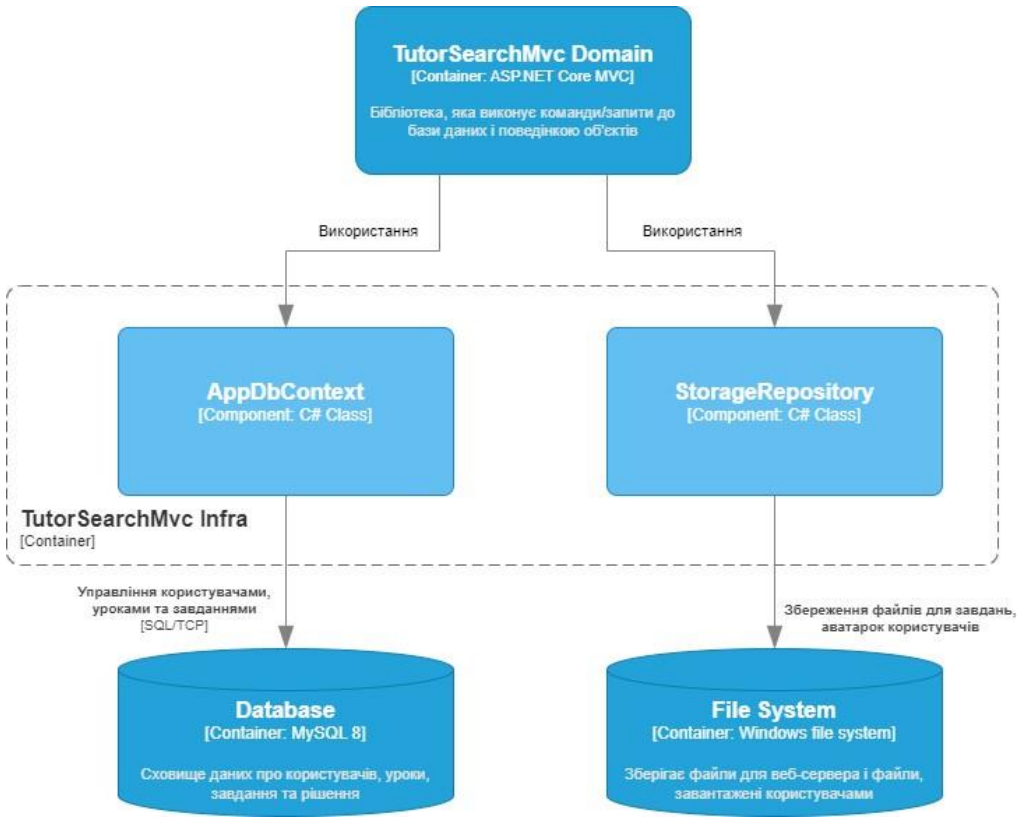


Рисунок 3.3 – Діаграма компонентів інфраструктурної бібліотеки

AppDbContext є компонентом, який відповідає за роботу з базою даних за допомогою Entity Framework Core (EF Core). Цей клас забезпечує контекст для доступу до таблиць бази даних та управління об'єктами. Він також наслідує від IdentityDbContext для управління користувачами та ролями.

StorageRepository є компонентом, що відповідає за роботу з файловою системою. Він надає методи для збереження, видалення та отримання файлів, а також для збереження аватарок користувачів.

У якості бази даних було обрано СУБД MySQL. До її переваг я можу віднести високу безпеку, оскільки MySQL є однією з найбезпечніших СУБД, яку використовують такі IT-гіганти, як Twitter, Facebook, та Wikipedia. Вона забезпечує зберігання паролів у зашифрованому вигляді за допомогою спеціальних алгоритмів шифрування, що гарантує надійний захист даних. Також MySQL відзначається чудовою масштабованістю, яка дозволяє системі адаптуватися до зростаючих навантажень, та високою продуктивністю MySQL, оскільки ця СУБД забезпечує обробку високошвидкісних транзакцій і надає можливість кешувати результати для пришвидшення повторних операцій. Налаштування індексів також сприяє

підвищенню продуктивності запитів до бази даних, а це є важливим для забезпечення ефективної роботи системи.

Взаємодію основних компонентів сутності домена CQRS зображено на рисунку 3.4.

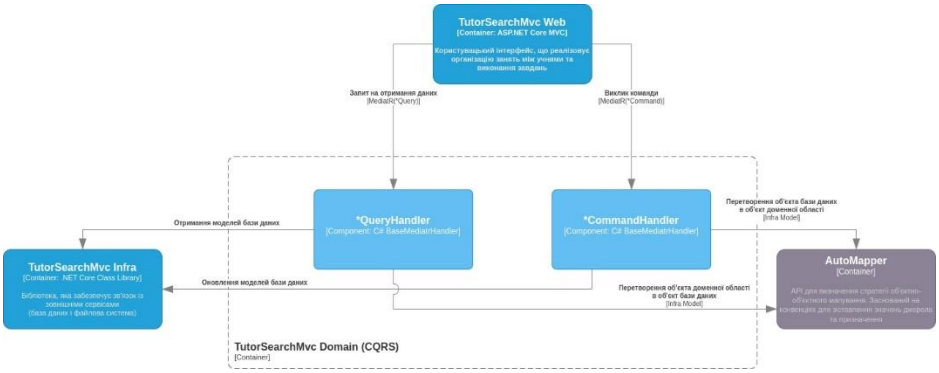


Рисунок 3.4 – Діаграма компонентів сутності домена CQRS

Взаємодію основних компонентів сутності доменної бібліотеки зображено на рисунку 3.5.

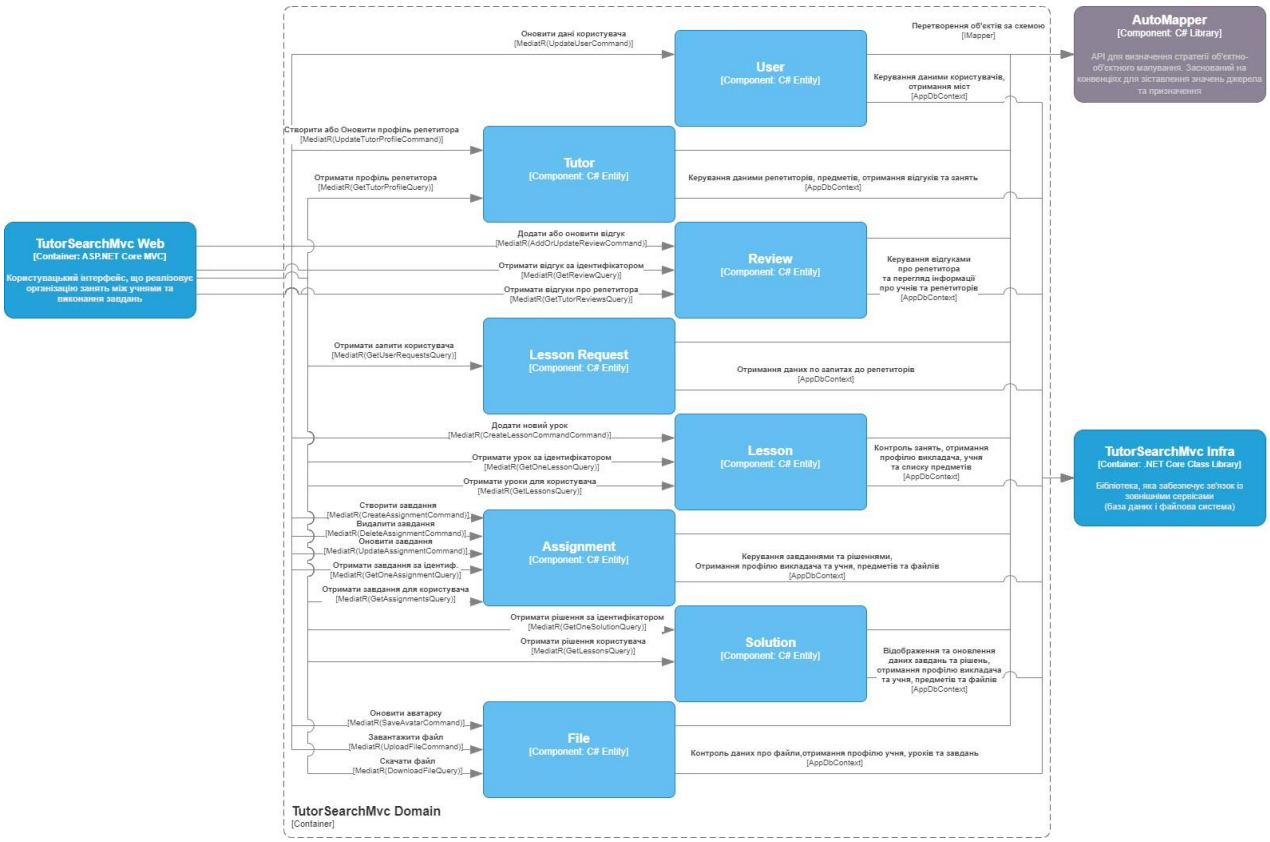


Рисунок 3.5 – Діаграма компонентів доменної бібліотеки

Взаємодію основних компонентів пошуку репетиторів зображено на рисунку 3.6.

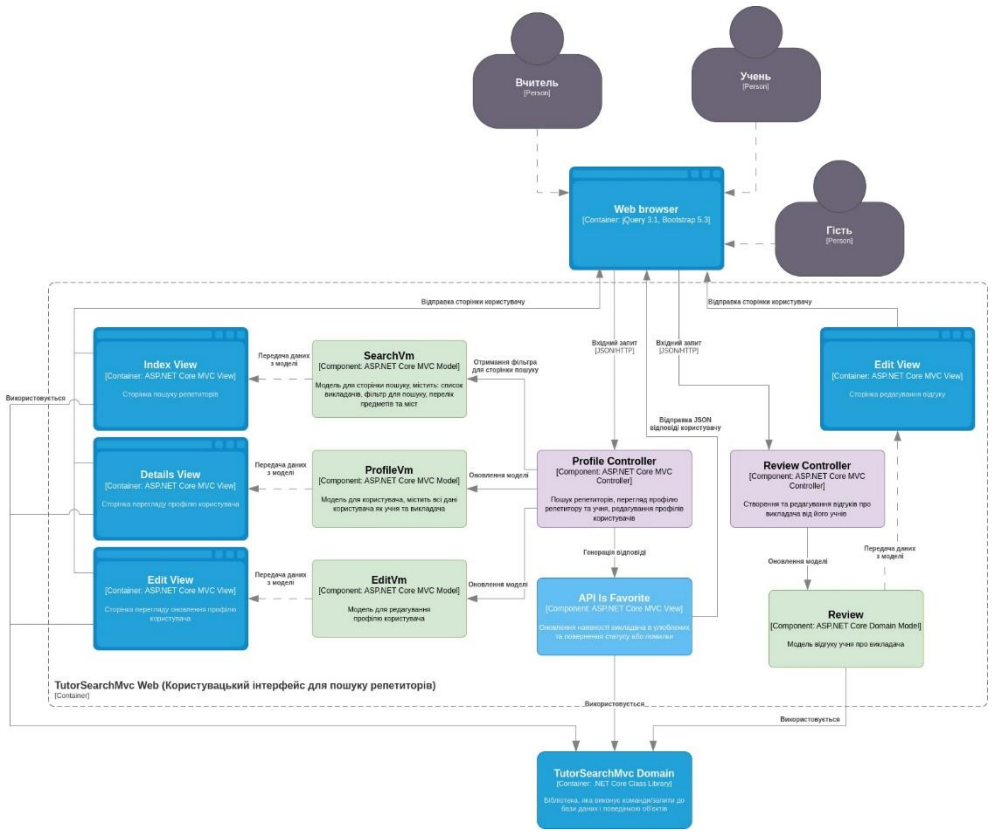


Рисунок 3.6 – Діаграма компонентів пошуку репетиторів

Взаємодію основних компонентів уроків зображено на рисунку 3.7.

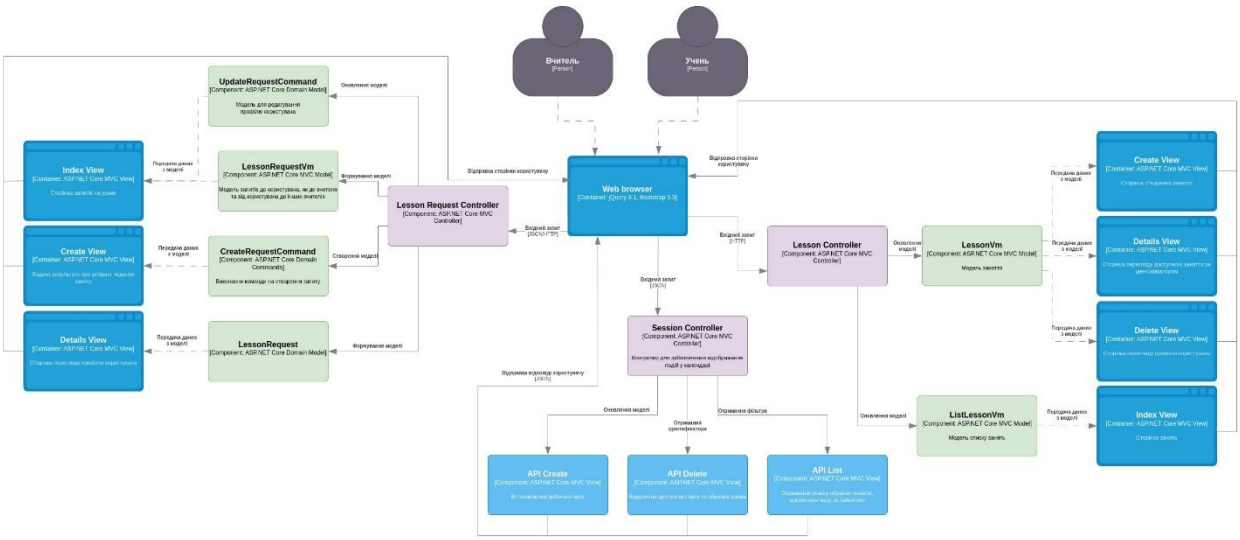


Рисунок 3.8 – Діаграма компонентів уроків

Взаємодію основних компонентів навчальних закладів зображено на рисунку 3.8.

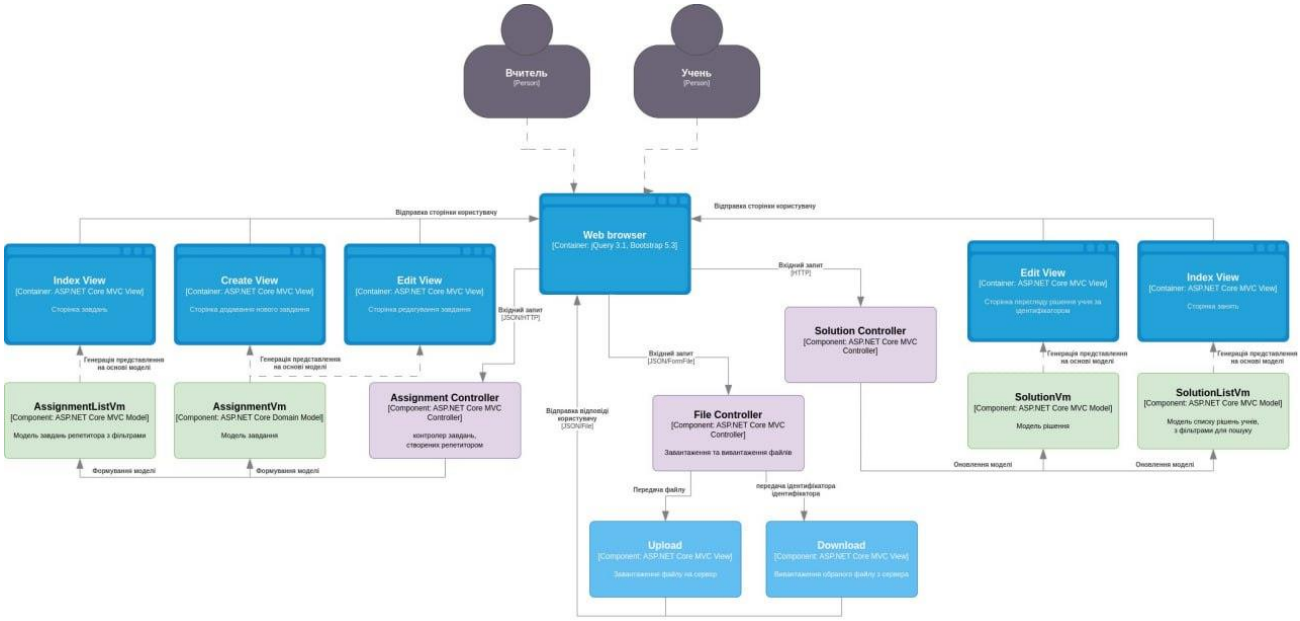


Рисунок 3.8 – Діаграма компонентів навчальних завдань

Стосовно авторизації та автентифікації користувачів було використано ASP.NET Core Identity[7], що є надійним фреймворком автентифікації та авторизації, який надається корпорацією Майкрософт для створення безпечних веб-додатків. Було обрано саме такий спосіб автентифікації та авторизації через надійне забезпечення безпеки даних користувачів. ASP.NET Core Identity забезпечує безпечне зберігання паролів користувачів, використовуючи односторонні алгоритми хешування, такі як bcrypt або PBKDF2. Завдяки цьому, навіть якщо хеші паролів потраплять до зловмисників, їх буде дуже складно змінити чи розшифрувати. Крім того ASP.NET Core Identity пропонує вбудовані інструменти для безпечного управління користувацькими даними. Він надійно зберігає інформацію про користувачів, включаючи їхні електронні адреси, ролі та паролі. Користувацькі дані зазвичай зберігаються у базі даних у захищеному вигляді, що убезпечує їх від загальних загроз безпеці, таких як SQL-ін'єкції. Також ASP.NET Core Identity забезпечує авторизацію на основі ролей. Це забезпечує,

користувачам доступ лише до тих ресурсів і дії, для яких вони мають авторизацію, що підвищує загальну безпеку програми. Завдяки цим функціям, ASP.NET Core Identity сприяє захисту користувацьких даних, надійному підтвердженню особи користувачів та запобіганню несанкціонованому доступу до конфіденційної інформації в веб-додатку.

3.2 Обґрунтування вибору засобів розробки

Проаналізуємо найчастіше використовувані мови розробки.

Python – це мова програмування з динамічною типізацією та високорівневий синтаксис, що акцентується на зрозумілості та легкості читання коду. Перевагами є широка колекція бібліотек для штучного інтелекту та машинного навчання, зрозумілий синтаксис, активна підтримка у галузях науки та аналізу даних. Однак у цієї мови програмування повільніше виконання порівняно з компільованими мовами, складне управління залежностями.

Java – це високорівнева мова програмування, орієнтована на об'єктно-орієнтований підхід, яка має значну популярність у веб-розробці. Її відомість базується на платформі JVM, яка дозволяє виконувати програми на будь-якій операційній системі без перекомпіляції для кожної окремо. Перевагами є кросплатформенність, широкий вибір додаткових бібліотек і фреймворків, високий рівень безпеки. Однак через використання віртуальної машини програми на Java можуть запускатися повільніше, ніж на деяких альтернативних платформах, наприклад, на .NET.

C# – це універсальна мова програмування з жорсткою типізацією, розроблена Microsoft для створення програмних продуктів на основі платформи .NET. .NET є відомою та ефективною платформою для веб-розробки, яка гарантує швидкодію та розмаїття можливостей для створення різних типів додатків. Перевагами є висока продуктивність, широка екосистема, що підтримується Microsoft. З недоліків можу виділити малу популярність на Linux для відкритих систем, можливість складнішого вивчення для новачків.

Проаналізувавши наведені вище мови програмування, було обрано для розробки платформу .NET. Для розробки повноцінного веб-застосунку було використано компонент .NET Core під назвою ASP.NET Core[8]. ASP.NET Core – відкрите програмне забезпечення, що використовується як основа для створення веб-застосунків. Стосовно баз даних в .NET Core є потужна бібліотека для взаємодії, це Entity Framework Core. Завдяки своїм функціональним можливостям, ця бібліотека надає зручний інтерфейс для роботи з базою даних, забезпечуючи швидкий та ефективний доступ до неї. Entity Framework Core спрощує процес взаємодії з базою даних, дозволяючи легко зв'язувати класи програмного коду з таблицями у базі даних. Крім того, ця бібліотека може автоматично створювати схему бази даних на основі класів, що полегшує та прискорює процес розробки. Незалежно від типу використовуваної системи керування базами даних, Entity Framework Core надає єдиний інтерфейс для взаємодії з будь-якою СУБД, що робить його надійним та популярним вибором серед багатьох розробників.

Для зберігання даних було обрано СУБД MySQL, основними перевагами якої є висока безпека даних та продуктивність.

Для клієнської частини було обрано мову програмування JavaScript. Для розмітки сторінки було використано HTML, для візуалізації – CSS.

3.3 Конструювання програмного забезпечення

ER діаграма сутностей зображена на рисунку 3.9.

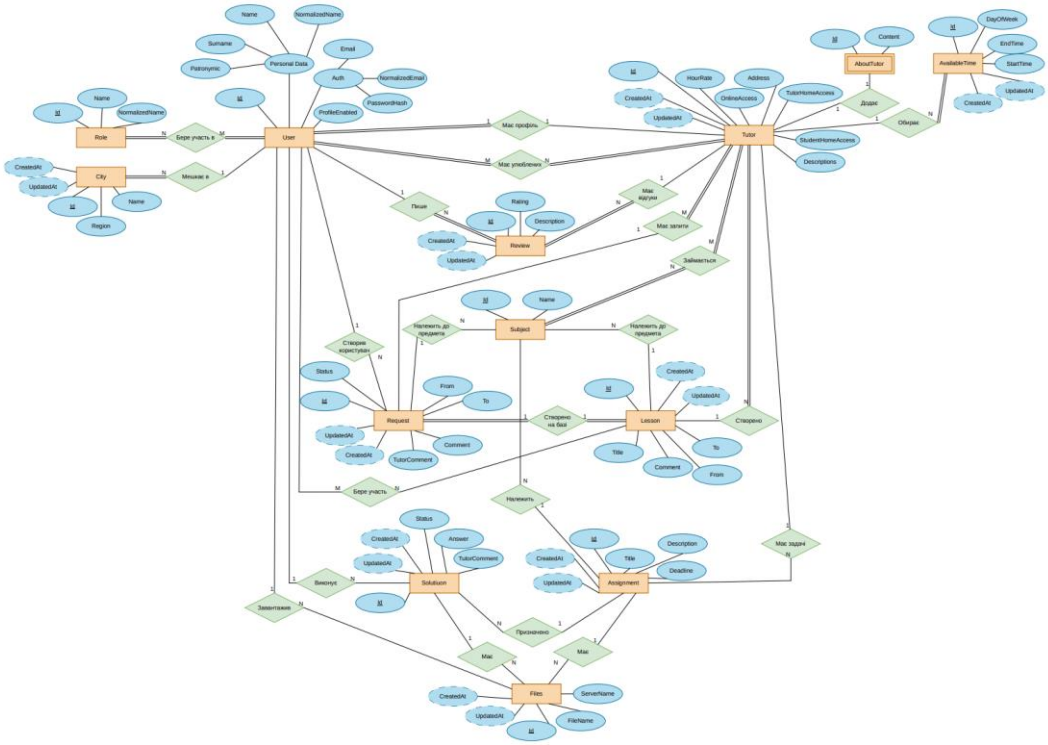


Рисунок 3.9 – ER діаграма сутностей User, Role, Tutor, Review, Subject, Request, Lesson, Solution, Assignment, Files, AboutTutor, AvailableTime

Як систему управління базами даних було використано MySQL. База даних серверу призначена для зберігання користувачів, а також даних про їхні уроки, завдання та відповіді. Опис таблиць бази даних наведено у таблицях 3.1 – 3.17. Логічна модель бази даних наведена на рисунку 3.10.

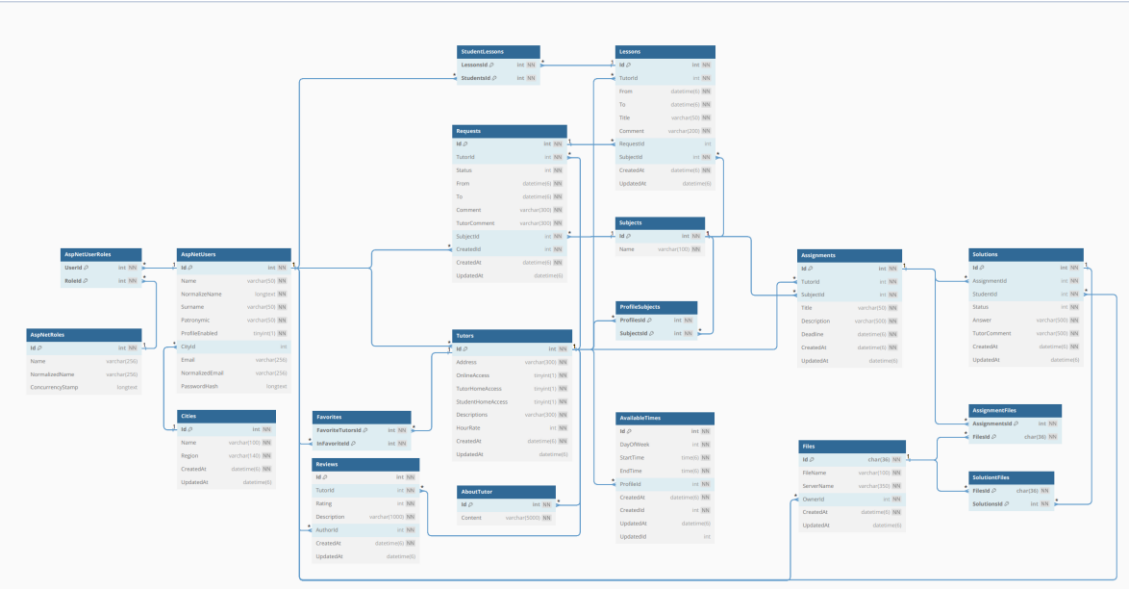


Рисунок 3.10 – Модель бази даних

Таблиця 3.1 – Опис таблиці AspNetUsers

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер користувача
Name	string	Ім'я користувача
Surname	string	Прізвище користувача
Patronymic	string	По батькові користувача
ProfileEnabled	bool	Вказує, чи активний профіль користувача
CityId	int	Ідентифікаційний номер міста користувача
Email	string	Електронна пошта користувача
NormalizedEmail	string	Нормалізована електронна пошта
PasswordHash	string	Хеш пароля користувача

Продовження таблиці 3.1

NormalizeName	string	Нормалізоване ім'я користувача

Таблиця 3.2 – Опис таблиці Cities

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер міста
Name	string	Назва міста
Region	string	Область міста
CreatedAt	datetime	Дата створення запису
UpdatedAt	datetime	Дата останнього оновлення запису

Таблиця 3.3 – Опис таблиці Favorites

Назва поля	Тип даних	Опис
FavoriteTutorsId	int	Ідентифікаційний номер користувача (улюбленого репетитора)
InFavoriteId	int	Ідентифікаційний номер користувача (який обрав)

Таблиця 3.4 – Опис таблиці Reviews

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер відгуку
TutorId	int	Ідентифікаційний номер репетитора
Rating	int	Рейтинг (0-10)

Продовження таблиці 3.4

Description	string	Опис відгуку
AuthorId	int	Ідентифікаційний номер автора відгуку
CreatedAt	datetime	Дата створення відгуку
UpdatedAt	datetime	Дата останнього оновлення відгуку

Таблиця 3.5 – Опис таблиці StudentLessons

Назва поля	Тип даних	Опис
LessonId	int	Ідентифікаційний номер уроку
StudentId	int	Ідентифікаційний номер студента

Таблиця 3.6 – Опис таблиці Requests

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер запиту
TutorId	int	Ідентифікаційний номер репетитора
Status	int	Статус запиту
From	datetime	Дата та час початку уроку
To	datetime	Дата та час закінчення уроку
Comment	string	Коментар до запиту
TutorComment	string	Посилання на заняття
SubjectId	int	Ідентифікаційний номер предмета

Продовження таблиці 3.6

CreatedId	int	Ідентифікаційний номер користувача, що створив запит
CreatedAt	datetime	Дата створення запиту
UpdatedAt	datetime	Дата останнього оновлення запиту

Таблиця 3.7 – Опис таблиці Requests

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер репетитора
Address	string	Адреса репетитора
OnlineAccess	bool	Вказує, чи доступний репетитор онлайн
TutorHomeAccess	bool	Вказує, чи доступний репетитор на дому
StudentHomeAccess	bool	Вказує, чи доступний репетитор для виїзду до студента
Descriptions	string	Опис репетитора
HourRate	int	Годинна ставка репетитора
CreatedAt	datetime	Дата створення запису
UpdatedAt	datetime	Дата останнього оновлення запису

Таблиця 3.8 – Опис таблиці AboutTutor

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер запису

Продовження таблиці 3.8

Content	string	Зміст запису про репетитора

Таблиця 3.9 – Опис таблиці Lessons

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер уроку
TutorId	int	Ідентифікаційний номер репетитора
From	datetime	Дата та час початку уроку
To	datetime	Дата та час закінчення уроку
Title	string	Назва уроку
Comment	string	Коментар до уроку
RequestId	int	Ідентифікаційний номер запиту
SubjectId	int	Ідентифікаційний номер предмета
CreatedAt	datetime	Дата створення уроку
UpdatedAt	datetime	Дата останнього оновлення уроку

Таблиця 3.10 – Опис таблиці Subjects

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер предмета
Name	string	Назва предмета

Таблиця 3.11 – Опис таблиці ProfileSubjects

Назва поля	Тип даних	Опис
ProfilesId	int	Ідентифікаційний номер профілю
SubjectsId	int	Ідентифікаційний номер предмета

Таблиця 3.12 – Опис таблиці AvailableTimes

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер запису
DayOfWeek	int	День тижня
StartTime	timeonly	Час початку
EndTime	timeonly	Час закінчення
ProfileId	int	Ідентифікаційний номер профілю
CreatedAt	datetime	Дата створення запису
CreatedId	int	Ідентифікаційний номер користувача, що створив запис
UpdatedAt	datetime	Дата останнього оновлення запису
UpdatedId	int	Ідентифікаційний номер користувача, що оновив запис

Таблиця 3.13 – Опис таблиці Assignments

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер завдання

Продовження таблиці 3.13

TutorId	int	Ідентифікаційний номер репетитора
SubjectId	int	Ідентифікаційний номер предмета
Title	string	Назва завдання
Description	string	Опис завдання
Deadline	datetime	Кінцевий термін виконання завдання
CreatedAt	datetime	Дата створення завдання
UpdatedAt	datetime	Дата останнього оновлення завдання

Таблиця 3.14 – Опис таблиці Files

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер файлу
FileName	string	Ім'я файлу
ServerName	string	Ім'я на сервері
OwnerId	int	Ідентифікаційний номер власника файлу
CreatedAt	datetime	Дата створення
UpdatedAt	datetime	Дата оновлення

Таблиця 3.15 – Опис таблиці Solutions

Назва поля	Тип даних	Опис
Id	int	Ідентифікаційний номер рішення
AssignmentId	int	Ідентифікаційний номер завдання

Продовження таблиці 3.15

Status	int	Статус рішення
Answer	string	Відповідь студента
TutorComment	string	Коментар репетитора
StudentId	int	Ідентифікаційний номер студента
CreatedAt	datetime	Дата створення
UpdatedAt	datetime	Дата останнього оновлення (може бути

Таблиця 3.16 – Опис таблиці AssignmentFiles

Назва поля	Тип даних	Опис
AssignmentsId	int	Ідентифікаційний номер завдання
FilesId	int	Ідентифікаційний номер файлу

Таблиця 3.17 – Опис таблиці SolutionFiles

Назва поля	Тип даних	Опис
SolutionsId	int	Ідентифікаційний номер рішення
FilesId	int	Ідентифікаційний номер файлу

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблицях 3.18-3.19.

Таблиця 3.18 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Microsoft Visual Studio	Головне середовище розробки програмного забезпечення серверної частини роботи.
		Інтегроване середовище розробки (IDE) для розробки вебзастосунків на JavaScript, TypeScript, HTML та CSS.
	MySQL Workbench	Програмне забезпечення, що надає легкий графічний інтерфейс для доступу до бази даних.

Таблиця 3.19 – Опис бібліотек серверної частини

№ п/п	Назва бібліотеки	Опис застосування
1	AutoMapper	Бібліотека для автоматичного перетворення об'єктів з одного типу в інший, що полегшує процес мапінгу між моделями даних.
2	MediatR	Реалізація шаблону посередника (Mediator), що забезпечує взаємодію між об'єктами без їх прямої залежності один від одного.

Продовження таблиці 3.19

№ п/п	Назва бібліотеки	Опис застосування
3	AspNetCore.Identity.EntityFrameworkCore	Інтеграція системи управління користувачами ASP.NET Core Identity з Entity Framework Core для роботи з базами даних.
	AspNetCore.EntityFrameworkCore.Tools	Інструменти для роботи з Entity Framework Core в ASP.NET Core, включаючи міграції та генерацію моделей з баз даних.
5	MySql.EntityFrameworkCore	Провайдер для використання MySQL з Entity Framework Core, що дозволяє працювати з базами даних MySQL в ASP.NET Core.
6	AspNetCore.Mvc.ViewFeatures	Бібліотека, що забезпечує підтримку різних функцій представлення у MVC додатках, таких як валідація моделей та TempData.
7	EntityFrameworkCore	ORM (Object-Relational Mapper) для .NET, який дозволяє взаємодіяти з базами даних за допомогою об'єктно-орієнтованого підходу.

Продовження таблиці 3.19

№ п/п	Назва бібліотеки	Опис застосування
8	EntityFrameworkCore.Design	Інструменти для розробки з використанням Entity Framework Core, включаючи підтримку командної стрічки та дизайнера моделей.
9	Extensions.DependencyInjection.Abstraction	Набір абстракцій для побудови контейнерів впровадження залежностей в NET, що сприяє легкій конфігурації залежностей.
10	SixLabors.ImageSharp	Бібліотека для роботи з зображеннями, яка дозволяє виконувати операції редагування та обробки зображень у .NET додатках.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Аналіз безпеки даних є важливим етапом у розробці будь-якого програмного забезпечення. Для забезпечення захисту даних в системі був прийнятий ряд заходів.

Одним з важливих аспектів є хешування паролей. За допомогою ASP.NET Core Identity пароль користувача зберігається у хеш-форматі, фактичний пароль залишається захищеним навіть у разі порушення бази даних, що знижує безпеку облікового запису.

Для захисту від впровадження система запобігає можливості введення SQL шляхом написання запитів SQL, зокрема параметризованих запитів, що

унеможливлює злом бази даних із введеними Користувачем даними через вразливість у коді.

Безпечне зберігання даних користувача забезпечується вбудованими функціями ASP.NET Core Identity, які надійно зберігають інформацію про користувачів, таку як адресу електронної пошти, ролі та паролі облікових записів. Дані користувачів зберігаються у базі даних у безпечному вигляді та захищені від SQL-ін'єкції.

Авторизація на основі ролей дозволяє користувачам отримувати доступ до ресурсів і виконувати тільки ті дії, які їм дозволені, тим самим підвищуючи безпеку програми.

Забезпечення наскрізної безпеки здійснюється за допомогою платформи Aiven, яка використовує виділені віртуальні машини, шифрування під час передачі та в стані спокою, а також автоматичні оновлення безпеки. Це дозволяє забезпечити високий рівень захисту даних на всіх етапах їх обробки та зберігання, мінімізуючи ризики компрометації.

Загалом, заходи, реалізовані в системі, включають хешування паролів, захист від ін'єкцій, безпечне зберігання даних користувачів, авторизацію на основі ролей, комплексний захист. Ці процедури сприяють всебічному захисту даних користувачів та запобігають несанкціонованому доступу до конфіденційної інформації.

Висновки до розділу

У третьому розділі було спроектовано архітектуру проекту з використання моделі C4, яка дозволяє зручно розділити систему на рівні контексту, контейнерів та компонентів, що сприяє її зрозумілості.

На рівні Domain були спроектовані основні компоненти бізнес-логіки, а саме велика кількість запитів на запис та читання, що були реалізовані з використанням підходу CQRS. На рівні Infrastructure були спроектовані компоненти для роботи з базою даних. На рівні вебу були спроектовані компонети патерну MVC, за допомогою яких користувачі можуть користуватись розробленим продуктом.

Вибір технологій та засобів розробки для вебзастосунку був обгрунтований їхніми перевагами та відповідністю вимогам проекту. Для розробки було обрано платформу .NET із використанням мови C# через її високу продуктивність, широку екосистему та підтримку з боку Microsoft. Взаємодія з базами даних здійснюється через потужну бібліотеку Entity Framework Core, що спрощує процес роботи з даними та забезпечує швидкий доступ до них. Для зберігання даних було обрано MySQL, що характеризується високою безпекою та продуктивністю. Клієнтська частина вебзастосунку створена з використанням JavaScript для динамічності, HTML для розмітки та CSS для стилізації.

Було представлено логічну модель бази даних та описано її поля.

Останнім етапом розділу був аналіз безпеки даних, який підтвердив, що створене програмне забезпечення відповідає актуальним стандартам якості та безпеки.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Статичне тестування є процесом аналізу програмного забезпечення, що включає аналіз коду без його виконання. Зазвичай включає в себе ревізію коду, перевірку стилю коду та тестування вимог до системи. Перевагами даного методу є виявлення багів на ранніх етапах циклу розробки та підвищення інформованості про проблеми якості.

Динамічне тестування є процесом аналізу програмного забезпечення шляхом виконання програми. Перевагою цього процесу є ретельне дослідження, яке розглядає всю функціональність програми, однак динамічне тестування займає багато часу та виконується після завершення кодування.

Для аналізу коду було обрано платформу статичного аналізу коду Embold[9], що виявляє структурні проблеми та витоки безпеки у кодовій базі на ранній стадії розробки. Аналіз якості коду на платформі Embold відображено на рисунку 4.1.

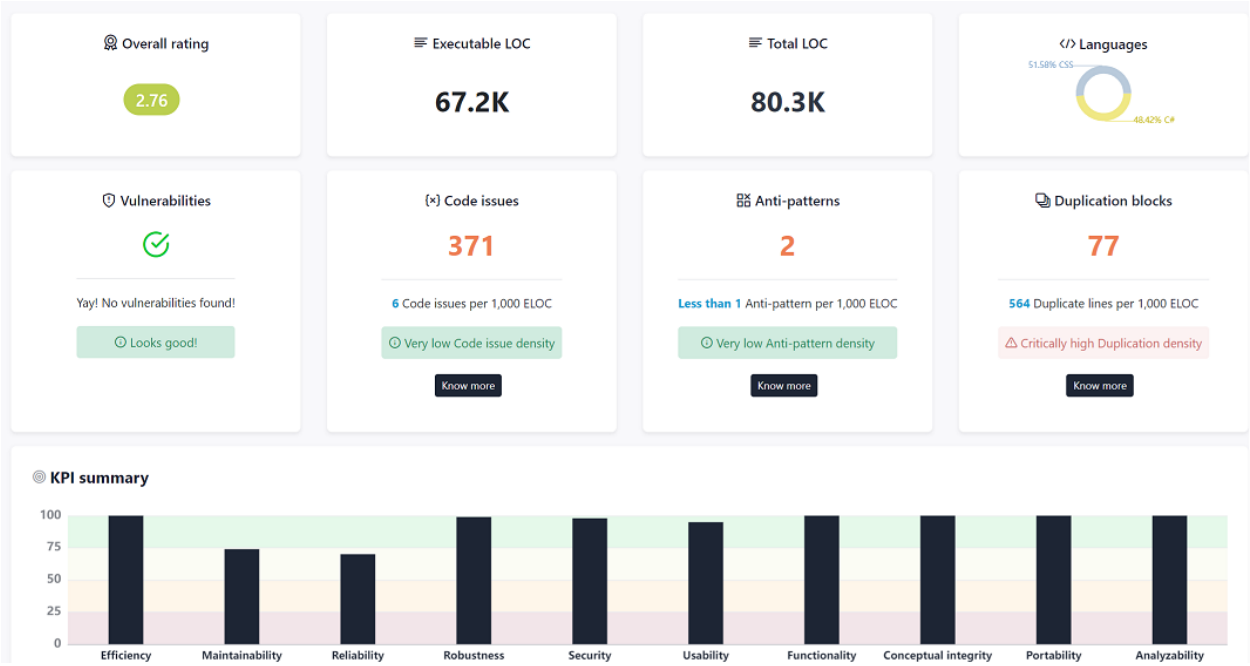


Рисунок 4.1 – Аналіз якості коду на платформі Embold

Як бачимо, під час аналізу вразливостей не було виявлено, що свідчить про відсутність загроз, майже не виявлено антишаблонів, незначна кількість проблем з кодом, що свідчить про гарну якість коду. Загальний рейтинг було отримано 2.76 від -5 до 5, що також вказує на непоганий результат. Також на рисунку 4.1 у частині KPI summary відображено ключові показники ефективності у вигляді діаграми результативності. Розглянемо основні показники KPI.

Ефективність є мірою того це міра, що визначає, наскільки швидко та точно можна виявити й виправити помилки, оптимізувати продуктивність та забезпечити безпеку застосунку. Вона враховує такі аспекти, як зручність читання коду та чіткість логіки. На діаграмі можна побачити високий показник у колонці Efficiency.

Функціональність є мірою, що визначає відповідність програмного забезпечення очікуванням користувачів та вимогам бізнесу. Ця метрика є важливою завдяки сприянню оцінці чи відповідає програмне забезпечення своєму функціоналу. На діаграмі можна побачити високий показник у колонці Functionality.

Підтримуваність є мірою, яка визначає чи легко можна змінити, адаптувати програмне забезпечення для вирішення проблем та виконання оновлень. Ця метрика є важливою завдяки сприянню оцінці стабільності та гнучкості системи, що є необхідними для забезпечення її довгострокової ефективності. На діаграмі можна побачити достаньо високий показник у колонці Maintainability.

Надійність є мірою, що визначає чи стабільно та безперебійно працює програмне забезпечення. Ця метрика є важливою завдяки сприянню оцінці можливості збоїв та відповіданню очікуванням користувачів щодо його стабільності та доступності. На діаграмі можна побачити достатньо високий показник у колонці Reliability.

Стійкість є мірою, що визначає здатність програми працювати належним чином за різних умов та впоратися з неочікуваними ситуаціями, такими як перебої з інтернет-з’єднанням або неправильно введені дані користувачів у поля. Ця метрика є важливою завдяки сприянню оцінці можливості ризику збоїв, що є

необхідним для підвищення довіри користувачів до продукту. На діаграмі можна побачити високий показник у колонці Robustness.

Зручність використання є мірою, що визначає чи легко та ефективно користувачі можуть використовувати програмне забезпечення. Ця метрика є важливою завдяки сприянню покращення задоволеності користувачів та забезпеченню більшої продуктивності. На діаграмі можна побачити високий показник у колонці Usability.

Безпека є мірою, що визначає захист програмного забезпечення від зовнішніх загроз та внутрішніх помилок. Ця метрика є важливою завдяк сприянню оцінці чи захищені дані та інформація користувачів, а це підвищує довіру до продукту на ринку. На діаграмі можна побачити високий показник у колонці Security.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1 – 4.30.

Таблиця 4.1 – Тест 1.1 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	У необхідні поля вводяться: коректна електронна пошта, що до цього не використовувалась у системі, пароль, підтвердження паролю, яке співпадає з паролем, введеним вище. Після цього натискається кнопка “Зареєструватися”.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку пошуку.

Продовження таблиці 4.1

Фактичний результат	Збігається з очікуваним.

Таблиця 4.2 – Тест 1.2 Авторизація користувача

Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	У необхідні поля вводяться: електронна пошта, що до цього не використовувалась у системі, і пароль, що був використаний під час реєстрації. Після цього натискається кнопка “Авторизуватися”.
Очікуваний результат	Авторизація проходить успішно, користувач перенаправляється на сторінку пошуку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 1.3 Авторизація адміністратора

Початковий стан системи	Адміністратор знаходиться на сторінці авторизації.
Вхідні дані	Електронна пошта, пароль адміністратора
Опис проведення тесту	У необхідні поля вводяться: коректна електронна пошта та пароль адміністратора, що є дійсним. Після цього натискається кнопка “Авторизуватися”.
Очікуваний результат	Авторизація проходить успішно, адміністратор перенаправляється на сторінку пошуку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 1.4 Вихід із системи

Початковий стан системи	Користувач авторизований у вебзастосунку
Вхідні дані	-
Опис проведення тесту	Користувач натискає кнопку виходу. Користувач натискає кнопку підтвердження виходу.
Очікуваний результат	Користувач деавторизований і повертається до ролі неавторизованого користувача.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 1.5 Пошук репетиторів

Початковий стан системи	Користувач знаходиться на головній сторінці
Вхідні дані	Повне імя репетитора, опції фільтрації (предмет, вартість, тип занять, місто)
Опис проведення тесту	У поле пошуку вводиться стрічка чатисни повного ім’ я існуючого репетитора. Користувач обирає параметри фільтрації та натискає кнопку “Шукати”.
Очікуваний результат	Система відображає список репетиторів згідно з обраними параметрами фільтрації та показує користувачу тільки репетиторів, повне ім’я яких містить шукану стрічку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 1.6 Створення профілю репетитора

Початковий стан системи	Користувач авторизований у вебзастосунку та знаходиться на сторінці профілю

Продовження таблиці 4.6

Вхідні дані	Активація чекбоксу створення профілю. Дані профілю репетитора (предмети, тип занять, вартість, адреса, опис, інформація про себе, календар)
Опис проведення тесту	У необхідні поля вводяться інформація про себе, що не перевищує 3000 символів та не більше 50 символів, домашня адреса, що не перевищує 300 символів, тип заняття – у вчителя/онлайн/в учня, короткий опис, що не перевищує 254 символи, ціна за одну годину вказана в діапазоні від 10 до 9999, предмети обрані з випадаючого списка. Після цього натискається кнопка створення профілю.
Очікуваний результат	Профіль репетитора успішно створено і збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 1.7 Редагування профілю репетитора

Початковий стан системи	Користувач авторизований, має профіль репетитора та знаходиться на сторінці профілю.
Вхідні дані	Оновлені дані профілю репетитора
Опис проведення тесту	Змінюються дані профілю, а саме інформація про себе, що не перевищує 3000 символів та не більше 50 символів, домашня адреса, що не перевищує 300 символів, тип заняття – у вчителя/онлайн/в учня, короткий опис, що не перевищує 254 символи, ціна за одну годину вказана в діапазоні від 10 до 9999, предмети обрані з випадаючого списка, години роботи репетитора в календарі. Після цього натискається кнопка оновлення профілю.

Продовження таблиці 4.7

Очікуваний результат	Профіль репетитора успішно оновлено і дані збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 1.8 Перевірка відповіді учня

Початковий стан системи	Користувач авторизований та знаходиться на сторінці завдань
Вхідні дані	Відповідь учня, коментар, статус виконання
Опис проведення тесту	Користувач обирає завдання зі статусом “На перевірці”, переглядає його, залишає коментар та змінює статус.
Очікуваний результат	Відповідь учня перевірена, коментар збережено, статус оновлено.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 1.9 Перегляд профілю репетитора

Початковий стан системи	Користувач знаходиться на сторінці пошуку.
Вхідні дані	Ідентифікаційний номер
Опис проведення тесту	Користувач натискає на обраного репетитора на сторінці пошуку.
Очікуваний результат	Профіль репетитора відображається коректно з усіма обов’язковими даними.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 1.10 Створення запиту на урок

Початковий стан системи	Користувач знаходиться у профілі обраного репетитора.
Вхідні дані	Час заняття, коментар до уроку
Опис проведення тесту	Користувач обирає доступний час заняття, вводить коментар, що не має перевищувати 1000 символів, та натискає кнопку відправлення запиту.
Очікуваний результат	Запит на урок створено та збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.11 – Тест 1.11 Створення відгуку

Початковий стан системи	Користувач авторизований та знаходиться у профілі обраного репетитора.
Вхідні дані	Рейтинг, коментар
Опис проведення тесту	Користувач заповнює форму відгуку, обирає рейтинг та заповнює поле коментаря, що не має перевищувати 1000 символів.
Очікуваний результат	Відгук успішно створено і збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.12 – Тест 1.12 Редагування відгуку

Початковий стан системи	Користувач авторизований та знаходиться у профілі обраного репетитора.
Вхідні дані	Рейтинг, коментар

Продовження таблиці 4.12

Опис проведення тесту	Користувач змінює рейтинг та коментар у відгуку, натискає кнопку збереження.
Очікуваний результат	Відгук успішно змінено і збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.13 – Тест 1.13 Створення профілю користувача

Початковий стан системи	Користувач авторизований та знаходиться на сторінці профілю.
Вхідні дані	Ім'я, прізвище, по-батькові, місто, аватар
Опис проведення тесту	У необхідні поля вводяться ім’я, прізвище та по батькові, що не перевищують 50 символів, обирається місто з випадаючого списку та завантажується аватар. Після цього натискається кнопка створення профілю.
Очікуваний результат	Профіль користувача успішно змінено і збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.14 – Тест 1.14 Редагування профілю користувача

Початковий стан системи	Користувач авторизований та знаходиться на сторінці профілю.
Вхідні дані	Ім'я, прізвище, по-батькові, місто, аватар
Опис проведення тесту	Змінюються дані профілю, а саме ім’я, прізвище, по батькові, що не перевищують 50 символів, обирається місто з випадаючого списку та завантажується аватар. Після цього натискається кнопка створення профілю.

Продовження таблиці 4.14

Очікуваний результат	Профіль користувача успішно змінено і збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.15 – Тест 1.15 Створення завдання

Початковий стан системи	Користувач авторизований, має профіль репетитора, хоча б один урок з учнем та знаходиться на сторінці завдань.
Вхідні дані	Назва завдання, опис завдання, термін здачі, файл
Опис проведення тесту	У необхідні поля вводяться назва завдання, що не перевищує 50 символів та не більше 3 символів, опис завдання, що не перевищує 500 символів, термін здачі, що за замочуванням виставлений на сьогоднішню дату плюс 7 днів, та завантажується файл завдання. Після цього натискається кнопка створення завдання.
Очікуваний результат	Завдання успішно створено, файл завантажено і збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.16 – Тест 1.16 Редагування завдання

Початковий стан системи	Користувач авторизований, має профіль репетитора, хоча б одне створене завдання та знаходиться на сторінці завдань.
Вхідні дані	Назва завдання, опис завдання, термін здачі, файл

Продовження таблиці 4.16

Опис проведення тесту	Замінюються дані завдання, а саме назва завдання, що не перевищує 50 символів та не більше 3 символів, опис завдання, що не перевищує 500 символів, термін здачі, що за замочуванням виставлений на сьогоднішню дату плюс 7 днів, та завантажуються додаткові файли завдання за бажанням. Після цього натискається кнопка збереження.
Очікуваний результат	Завдання успішно змінено і збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.17 – Тест 1.17 Видалення завдання

Початковий стан системи	Користувач авторизований, має профіль репетитора та знаходиться на сторінці обраного завдання
Вхідні дані	Ідентифікаційний номер завдання
Опис проведення тесту	Користувач натискає кнопку видалення.
Очікуваний результат	Завдання успішно видалено із системи.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.18 – Тест 1.18 Створення відповіді

Початковий стан системи	Користувач авторизований та має виставлене завдання.
Вхідні дані	Відповідь, файл
Опис проведення тесту	У відповідне поле вводиться відповідь, що не перевищує 500 символів, та завантажується файл відповіді

Продовження таблиці 4.18

Очікуваний результат	Відповідь успішно створено, файл завантажено і збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.19 – Тест 1.19 Редагування відповіді

Початковий стан системи	Користувач авторизований та має відповідь на завдання.
Вхідні дані	Відповідь, файл
Опис проведення тесту	Змінюються дані відповіді, а саме поле відповіді, що не перевищує 500 символів, та завантажується файл відповіді
Очікуваний результат	Відповідь успішно змінено і збережено у системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.20 – Тест 1.20 Перегляд списку уроків

Початковий стан системи	Користувач авторизований та має хоча б один схвалений урок.
Вхідні дані	Натискання на посилання “Уроки” в навігаційному меню
Опис проведення тесту	Користувач з навігаційного меню переходить на сторінку “Уроки”. На сторінці уроків відображається список уроків із предметом, вчителем, учнем, датою, початком та кінцем заняття. Користувач обирає один із уроків для перегляду, після чого переглядає його та повертається до списку уроків.

Продовження таблиці 4.20

Очікуваний результат	Кожен урок із предметом, вчителем, учнем, датою, початком та кінцем заняття. Після вибору уроку відображається повна інформація про урок і користувач може повернутися до списку уроків.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.21 – Тест 1.21 Перегляд списку запитів

Початковий стан системи	Користувач авторизований та має хоча б один запит на заняття
Вхідні дані	Натискання на посилання “Запити” в навігаційному меню
Опис проведення тесту	Відображення списку запитів на сторінці із предметом, вчителем, учнем, датою, початком та кінцем заняття, статусом, посиланням на заняття для учня. Після вибору запиту відображається повна інформація про запит, користувач обирає статус запиту, якщо він є викладачем, і повертається до списку запитів.
Очікуваний результат	Кожен запит із предметом, вчителем, учнем, датою, початком та кінцем заняття, статусом, посиланням на заняття для учня. Після вибору уроку відображається повна інформація про запит, користувач оможе обрати статус запиту, якщо він є викладачем, і повернутися до списку запитів.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.22 – Тест 1.22 Створення заняття

Початковий стан системи	Користувач авторизований та має профіль репетитора
Вхідні дані	Дата, час початку та кінця, коментар, предмет, учень
Опис проведення тесту	У необхідні поля вводяться назва завдання, що не перевищує 50 символів та не менше за 3 символи, опис завдання, що не перевищує 500 символів, та завантажується файл завдання.
Очікуваний результат	Заняття успішно створено і збережено у системі
Фактичний результат	Збігається з очікуваним.

Таблиця 4.23 – Тест 1.23 Перегляд списку виставлених завдань

Початковий стан системи	Користувач авторизований та має хоча б одне завдання
Вхідні дані	Натискання на посилання “Завдання для виконання” в навігаційному меню
Опис проведення тесту	Відображення списку виставлених завдань на сторінці із предметом, назвою, вчителем, учнем, статусом.
Очікуваний результат	Кожне завдання із коректними предметом, назвою, вчителем, учнем, статусом.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.24 – Тест 1.24 Перегляд списку своїх завдань

Початковий стан системи	Користувач авторизований та створив хоча б одне завдання учню

Продовження таблиці 4.24

Вхідні дані	Натискання на посилання “Мої завдання” в навігаційному меню
Опис проведення тесту	Користувач з навігаційного меню переходить на сторінку “Мої завдання”. На сторінці завдань відображається список створених завдань користувачем із назвою завдання, терміном здачі та предметом викладання. Користувач може перейти на сторінку редагування завдання під час натискання кнопки редагування завдання або на сторінку виставлених завдань з відображенням обраного завдання під час натискання кнопки перевірки для перегляду статусу виконання.
Очікуваний результат	Завдання користувача на сторінці із коректними назвою завдання, терміном здачі завдання та предметом викладання.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.25 – Тест 1.25 Додавання предмету

Початковий стан системи	Адміністратор авторизований
Вхідні дані	Назва предмету
Опис проведення тесту	Адміністратор авторизований та переходить на сторінку списку предметів. У відповідне поле вводить назву предмету. Натискає кнопку збереження.
Очікуваний результат	Предмет успішно додано і збережено у системі
Фактичний результат	Збігається з очікуваним

Таблиця 4.26 – Тест 1.26 Додавання міста

Початковий стан системи	Адміністратор авторизований
Вхідні дані	Назва міста
Опис проведення тесту	Адміністратор авторизований та переходить на сторінку списку міст. У відповідне поле вводить назву міста. Натискає кнопку збереження.
Очікуваний результат	Місто успішно додано і збережено у системі
Фактичний результат	Збігається з очікуваним

Таблиця 4.27 – Тест 1.27 Редагування міста

Початковий стан системи	Адміністратор авторизований
Вхідні дані	Назва міста
Опис проведення тесту	Адміністратор авторизований та переходить на сторінку списку міст. Натискає кнопку редагування. Змінює назву міста. Натискає кнопку збереження.
Очікуваний результат	Місто успішно відредаговано і збережено у системі
Фактичний результат	Збігається з очікуваним

Таблиця 4.28 – Тест 1.28 Редагування предмету

Початковий стан системи	Адміністратор авторизований
Вхідні дані	Назва предмету

Продовження таблиці 4.28

Опис проведення тесту	Адміністратор авторизований та переходить на сторінку списку міст. Натискає кнопку редагування. Змінює назву предмету. Натискає кнопку збереження.
Очікуваний результат	Предмет успішно відредаговано і збережено у системі
Фактичний результат	Збігається з очікуваним

Таблиця 4.29 – Тест 1.29 Додавання репетитора в улюблені

Початковий стан системи	Користувач авторизований
Вхідні дані	Ідентифікатор репетитора
Опис проведення тесту	Користувач авторизований та переходить на сторінку профілю репетитора. Натискає кнопку додавання в улюблені.
Очікуваний результат	Репетитора успішно додано в улюблені та відображено на головній сторінці при натисканні кнопки улублених репетиторів
Фактичний результат	Збігається з очікуваним

Таблиця 4.30 – Тест 1.30 Додавання рейтингу репетитора

Початковий стан системи	Користувач авторизований і мав заняття з репетитором
Вхідні дані	Рейтинг (1-10)
Опис проведення тесту	Користувач авторизований, переходить на сторінку профілю репетитора, додає рейтинг

Продовження таблиці 4.30

Очікуваний результат	Рейтинг репетитора успішно оновлено
Фактичний результат	Збігається з очікуваним

Таблиця 4.31 – Тест 1.31 Створення відповіді на запит

Початковий стан системи	Користувач авторизований і має хоча б один запит
Вхідні дані	Посилання на урок, статус
Опис проведення тесту	У відповідне поле вводиться посилання на урок, що не перевищує 254 символи та не менше за 3 символи, опис завдання, що не перевищує 500 символів, та завантажується файл завдання.
Очікуваний результат	Відповідь на запит успішно створено
Фактичний результат	Збігається з очікуваним

Таблиця 4.32 – Тест 1.32 Перегляд уроку

Початковий стан системи	Користувач знаходиться на сторінці уроків.
Вхідні дані	Ідентифікаційний номер
Опис проведення тесту	Користувач натискає на обраний урок на сторінці уроків.
Очікуваний результат	Урок відображається коректно з усіма обов’язковими даними.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.33 – Тест 1.33 Видалення уроку

Початковий стан системи	Користувач авторизований, має хоча б один урок та знаходиться на сторінці уроку.
Вхідні дані	Ідентифікаційний номер заняття
Опис проведення тесту	Користувач натискає кнопку видалення.
Очікуваний результат	Урок успішно видалено із системи.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.34 – Тест 1.34 Видалення предмету

Початковий стан системи	Адміністратор авторизований та знаходиться на сторінці предмету.
Вхідні дані	Ідентифікаційний номер предмету
Опис проведення тесту	Адміністратор натискає кнопку видалення.
Очікуваний результат	Предмет викладання успішно видалено із системи.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.35 – Тест 1.35 Видалення міста

Початковий стан системи	Адміністратор авторизований та знаходиться на сторінці міста.
Вхідні дані	Ідентифікаційний номер міста
Опис проведення тесту	Адміністратор натискає кнопку видалення.
Очікуваний результат	Місто викладання успішно видалено із системи.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.36 – Тест 1.36 Видалення користувача

Початковий стан системи	Адміністратор авторизований та знаходиться на сторінці списку користувачів.
Вхідні дані	Ідентифікаційний номер користувача
Опис проведення тесту	Адміністратор натискає кнопку видалення.
Очікуваний результат	Профіль користувача успішно видалено із системи. Роль користувача змінено на видалену.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.37 – Тест 1.37 Видалення облікових даних

Початковий стан системи	Користувач авторизований та знаходиться на сторінці видаленого профілю.
Вхідні дані	Ідентифікаційний номер користувача
Опис проведення тесту	Користувач натискає кнопку видалення.
Очікуваний результат	Облікові дані користувача успішно видалено із системи.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

В якості контрольного прикладу було обрано створення, перегляд профілю та пошук профілей репетиторів. Дане мануальне тестування описано у таблицях 4.38-4.40.

Таблиця 4.38 – Тест 1.5 Пошук репетиторів

Початковий стан системи	Користувач знаходиться на головній сторінці

Продовження таблиці 4.38

Вхідні дані	Повне імя репетитора, опції фільтрації (предмет, вартість, тип занять, місто)
Опис проведення тесту	У поле пошуку вводиться стрічка частини повного ім’ я існуючого репетитора. Користувач обирає параметри фільтрації та натискає кнопку “Шукати”.
Очікуваний результат	Система відображає список репетиторів згідно з обраними параметрами фільтрації та показує користувачу тільки репетиторів, повне ім’я яких містить шукану стрічку.
Фактичний результат	Збігається з очікуваним.

Пошук репетитора за частиною повного ім’я відображено на рисунку 4.2.

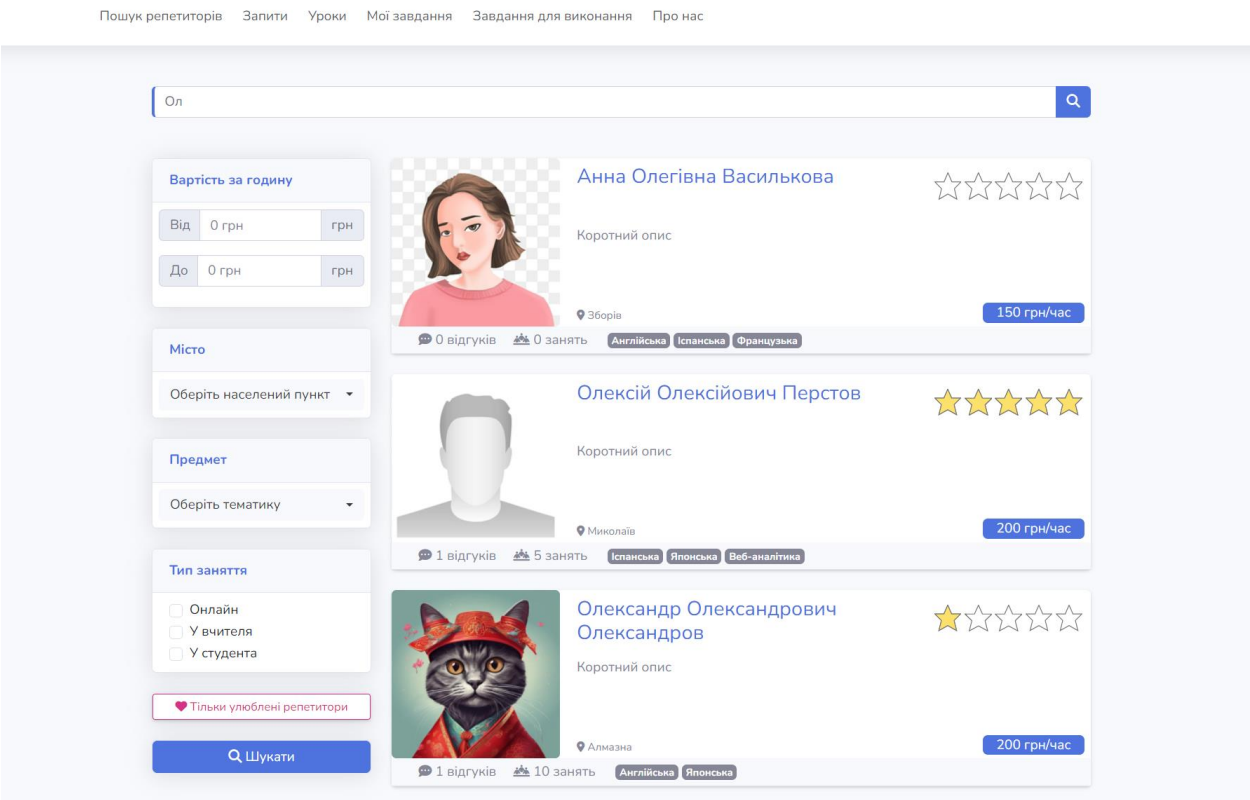


Рисунок 4.2 – Приклад пошуку репетитора за частиною повного ім’я

Пошук репетитора за фільтрацією “Вартість за годину” відображено на рисунку 4.3.

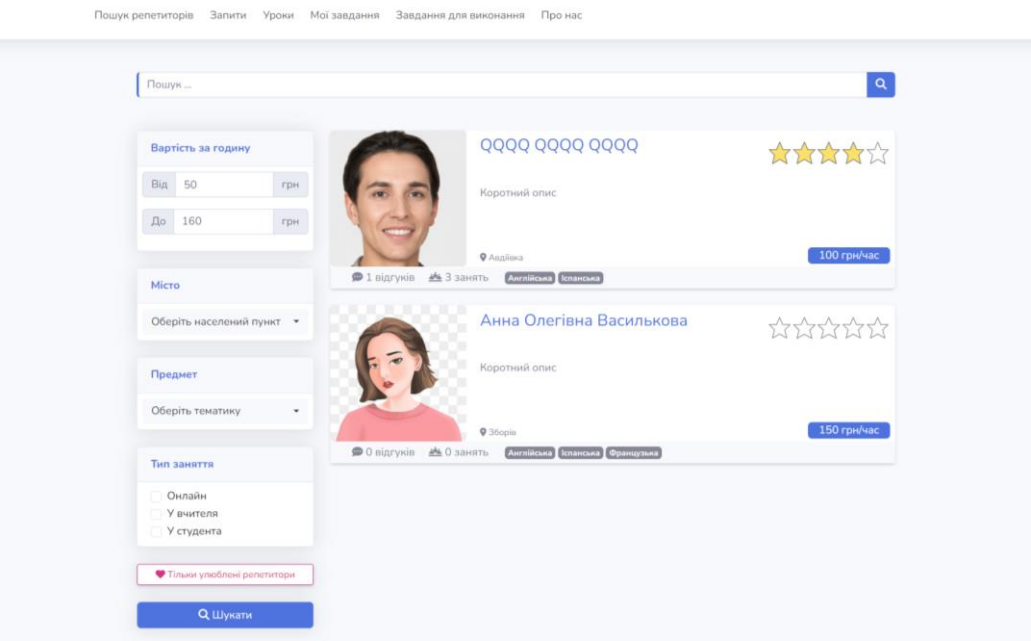


Рисунок 4.3 – Приклад пошуку репетитора за фільтрацією “Вартість за годину”

Можливе розгалудження:

Якщо значення вартості буде задано менше 0, що неможливо, користувач побачить повідомлення з рисунку 4.4.

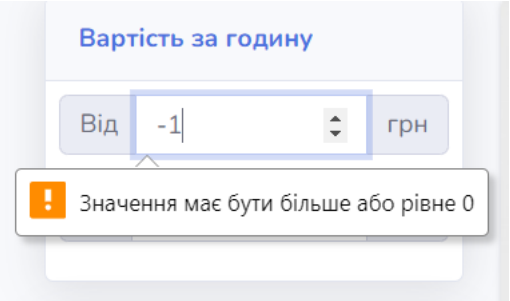


Рисунок 4.4 – Поля для ранжування вартості заняття

Пошук репетитора за фільтрацією “Місто” відображено на рисунку 4.4.

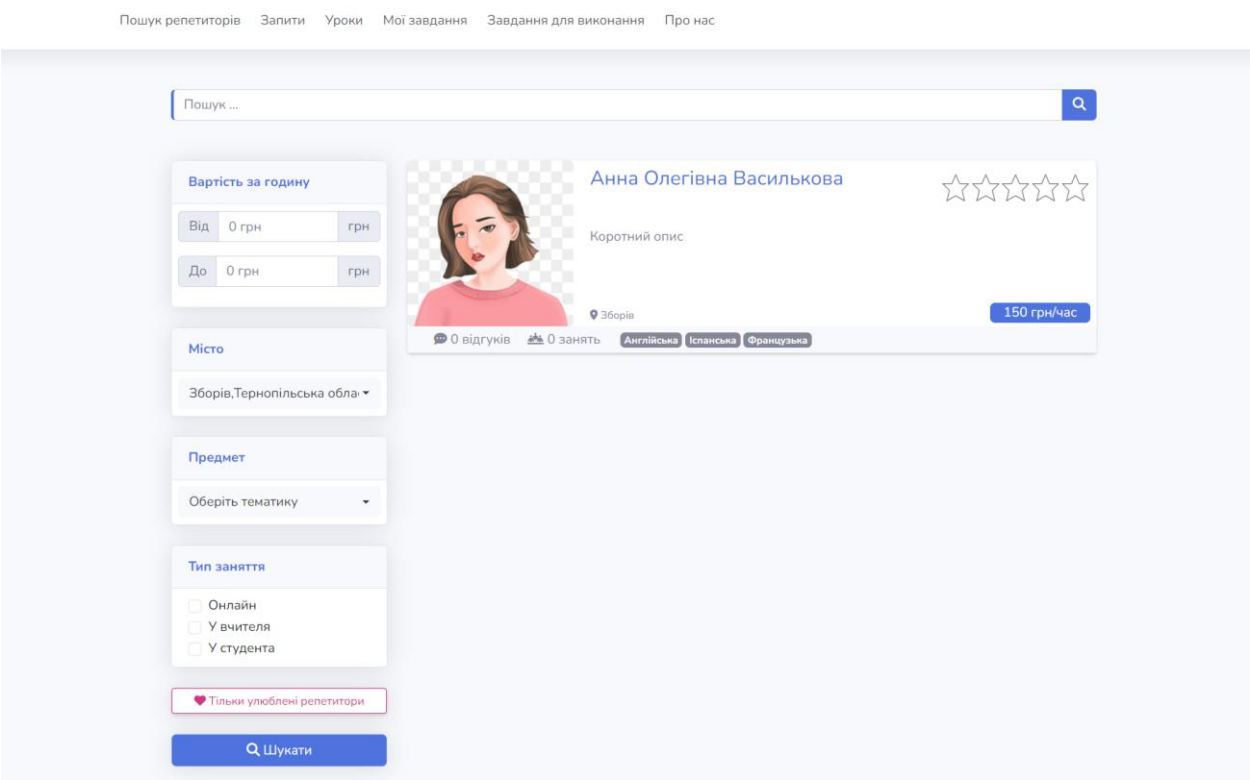


Рисунок 4.5 – Приклад пошуку репетитора за фільтрацією “Місто”

Пошук репетитора за фільтрацією “Предмет” відображено на рисунку 4.6.

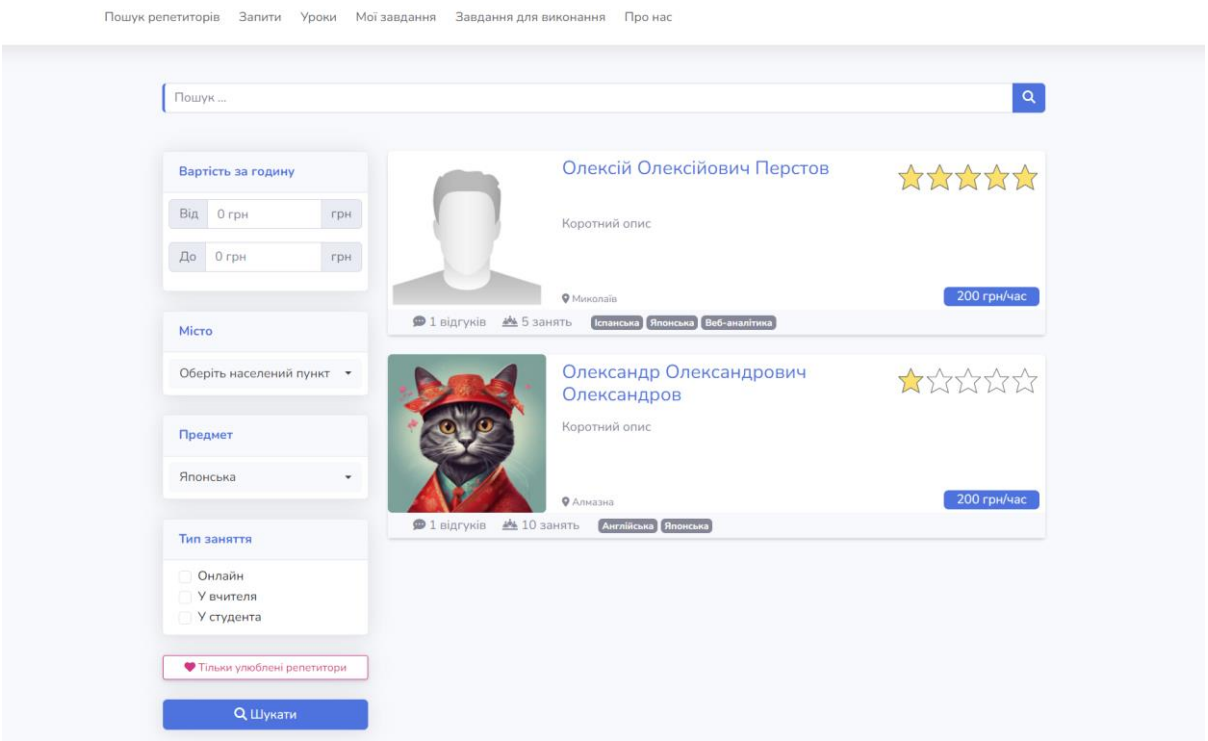


Рисунок 4.6 – Приклад пошуку репетитора за фільтрацією “Предмет”

Пошук репетитора за фільтрацією “Тип заняття” відображено на рисунку 4.7.

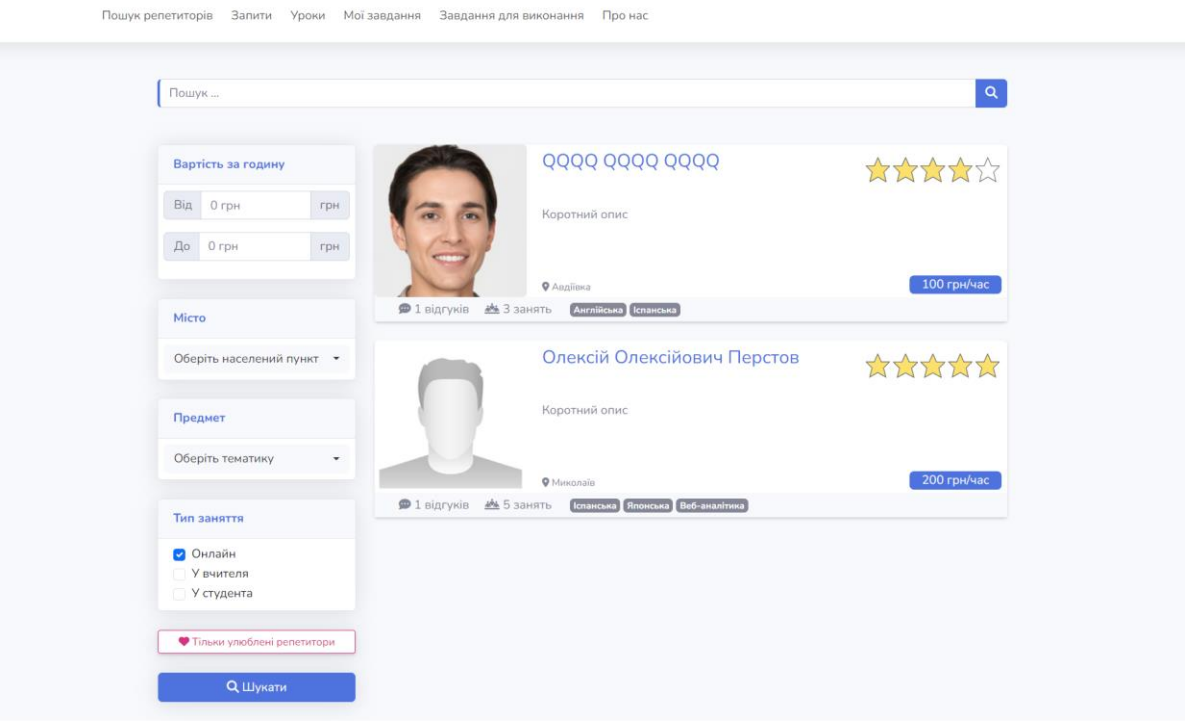


Рисунок 4.7 – Приклад пошуку репетитора за фільтрацією “Тип заняття”

Пошук репетитора за повним ім’ям та опціями фільтрації відображено на рисунку 4.8.

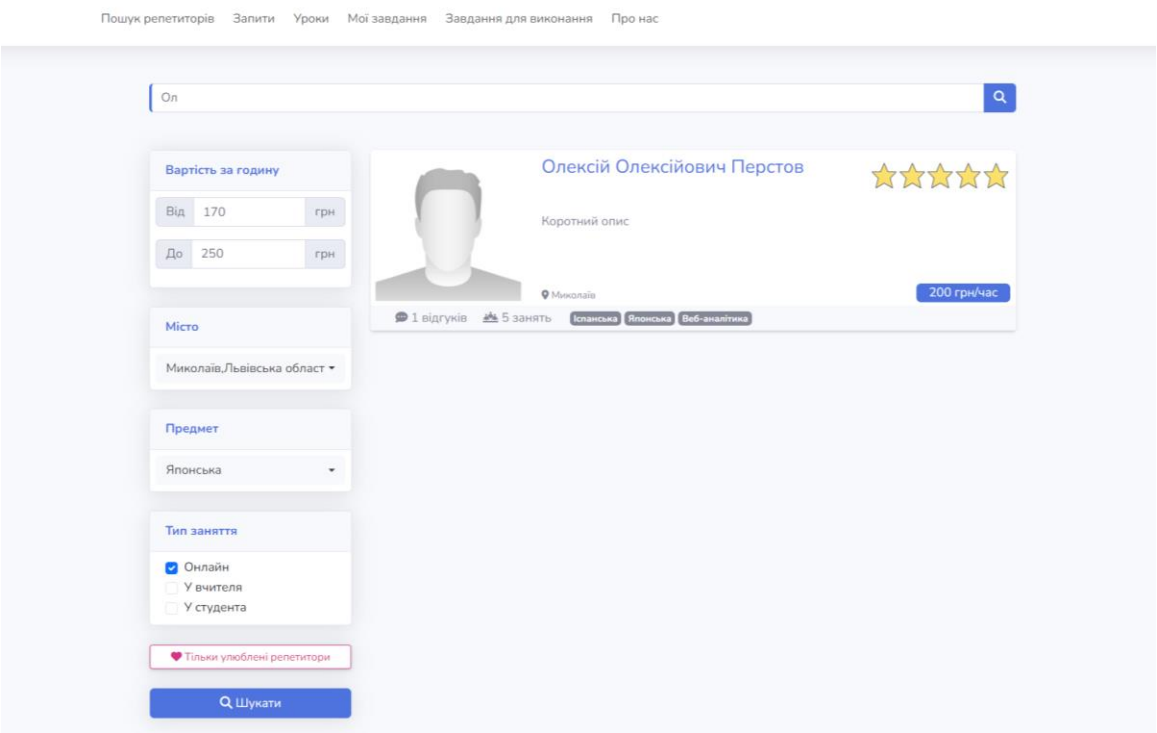


Рисунок 4.8 – Приклад пошуку репетитора за повним ім’ям та опціями фільтрації

Таблиця 4.39 – Тест 1.6 Створення профілю репетитора

Початковий стан системи	Користувач авторизований у вебзастосунку та знаходиться на сторінці профілю
Вхідні дані	Активація чекбоксу створення профілю. Дані профілю репетитора (предмети, тип занять, вартість, адреса, короткий опис, інформація про себе, календар).
Опис проведення тесту	У необхідні поля вводяться інформація про себе, що не перевищує 3000 символів та не більше 50 символів, домашня адреса, що не перевищує 300 символів, тип заняття – у вчителя/онлайн/в учня, короткий опис, що не перевищує 254 символи, ціна за одну годину вказана в діапазоні від 10 до 9999, предмети обрані з випадаючого списка. Після цього натискається кнопка створення профілю.
Очікуваний результат	Профіль репетитора успішно створено і збережено у системі.

Продовження таблиці 4.39

Фактичний результат	Збігається з очікуваним.

Створення профілю репетитора відображено на рисунках 4.9 - 4.10.

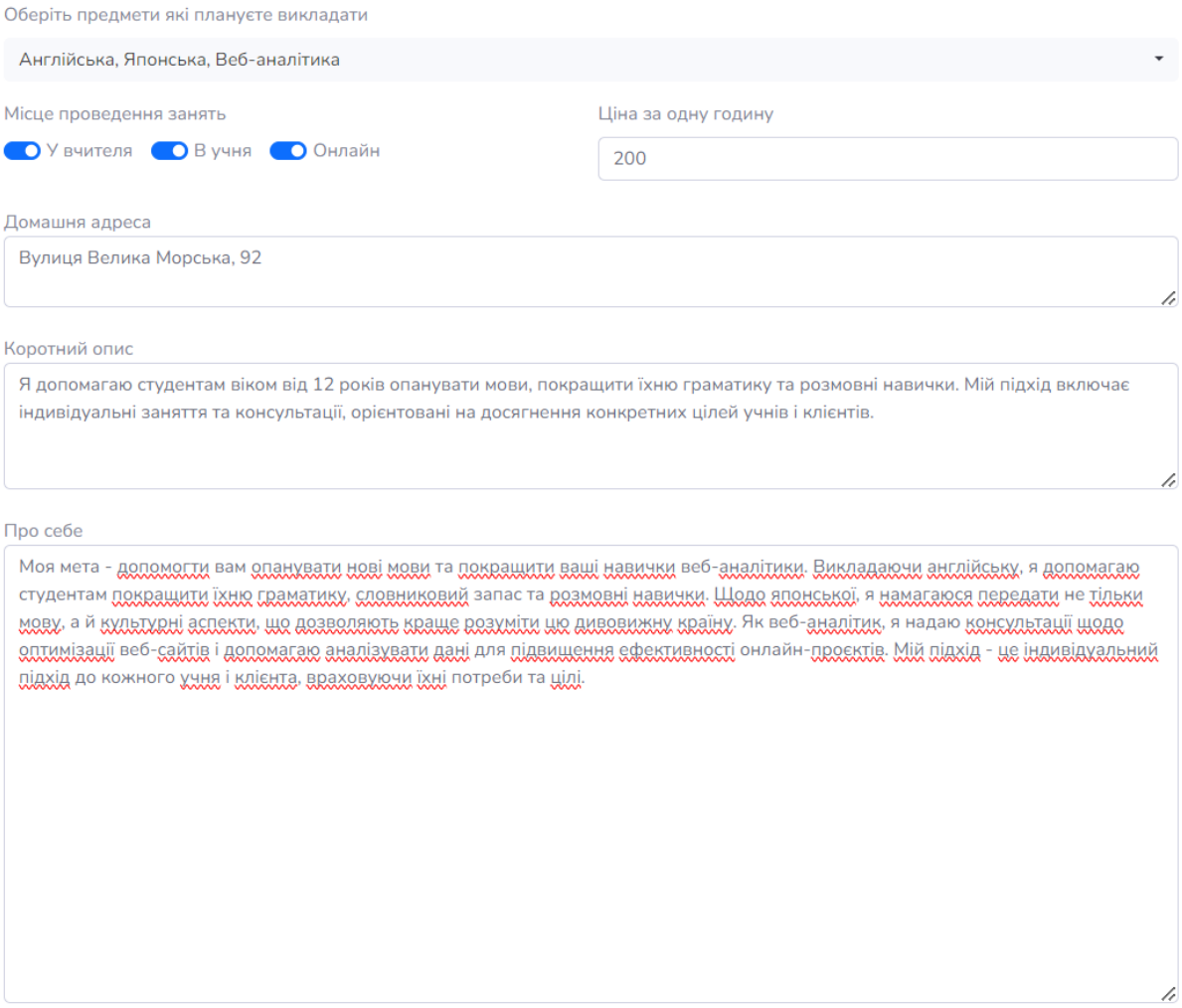


Рисунок 4.9 – Приклад створення профілю репетитора

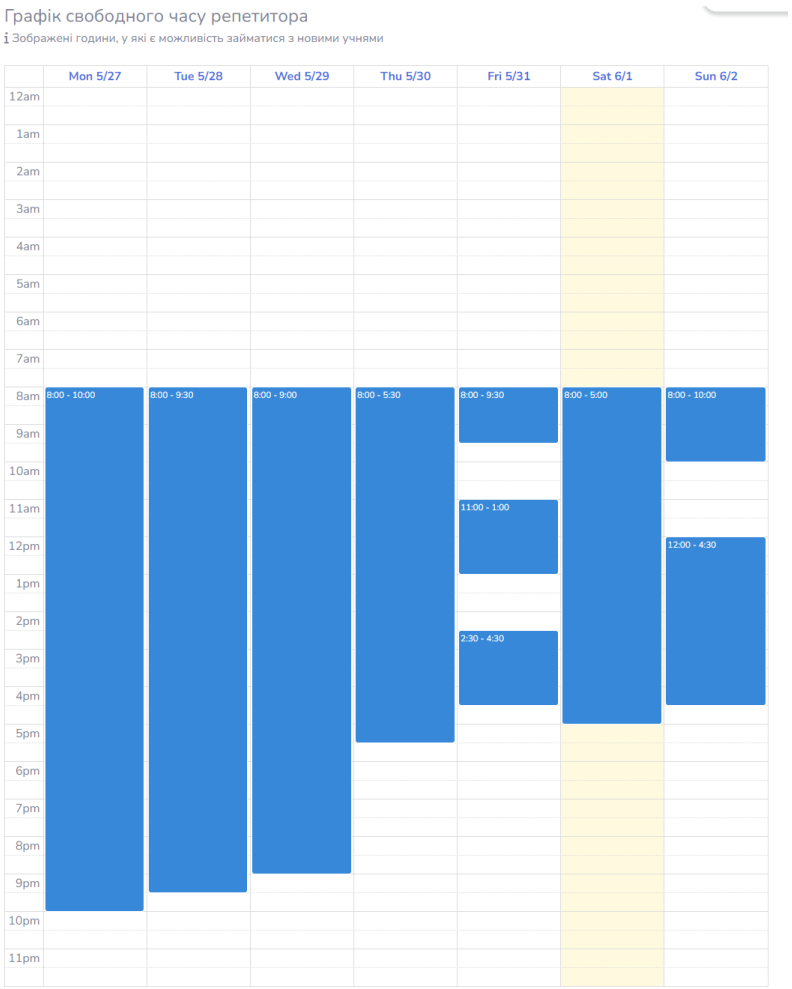


Рисунок 4.10 – Приклад календаря під час створення профілю

Можливі розгалудження:

Якщо значення вартості буде задано не в діапазоні від 0 до 9999, користувач побачить повідомлення з рисунку 4.11.

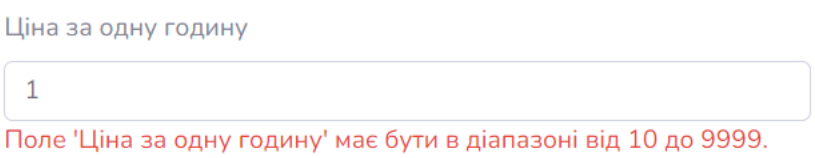


Рисунок 4.11 – Відображення повідомлення про не валідну вартість заняття

Якщо поля домашньої адреси, короткого опису чи про себе залишаться попрожніми, користувач побачить повідомлення з рисунку 4.12.

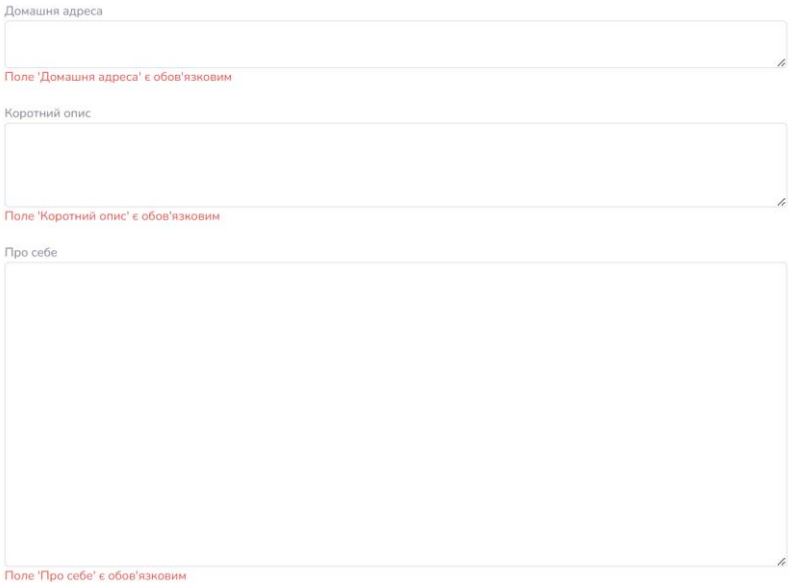


Рисунок 4.12 – Відображення повідомлень про порожні поля, що є обов’язковими для заповнення

Таблиця 4.40 – Тест 1.9 Перегляд профілю репетитора

Початковий стан системи	Користувач знаходиться на сторінці пошуку.
Вхідні дані	Ідентифікаційний номер
Опис проведення тесту	Користувач натискає на обраного репетитора на сторінці пошуку.
Очікуваний результат	Профіль репетитора відображається коректно з усіма обов’язковими даними.
Фактичний результат	Збігається з очікуваним.

Перегляд профілю репетитора відображено на рисунках 4.13 – 4.14.

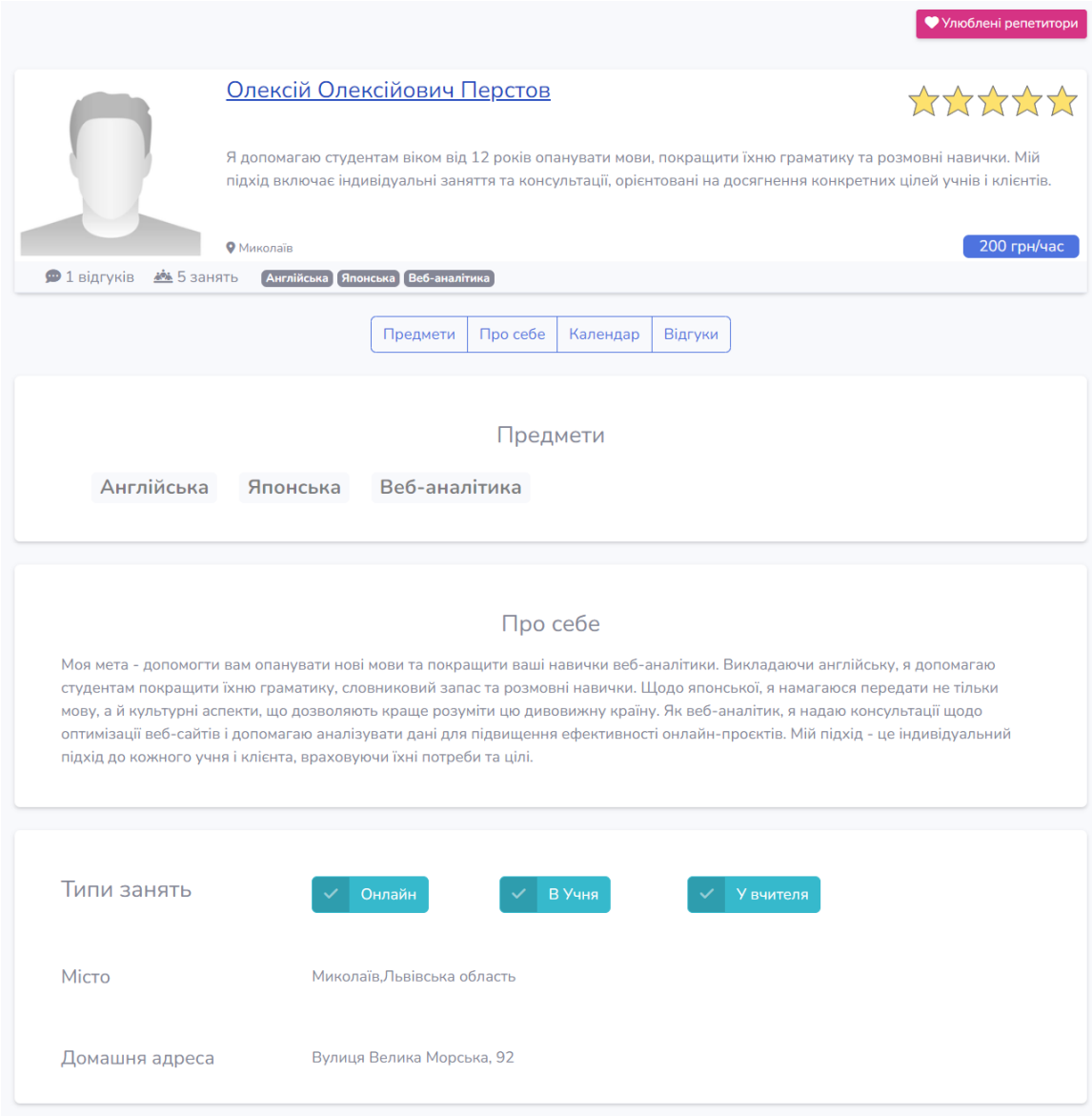


Рисунок 4.13 – Приклад відображення профілю репетитора

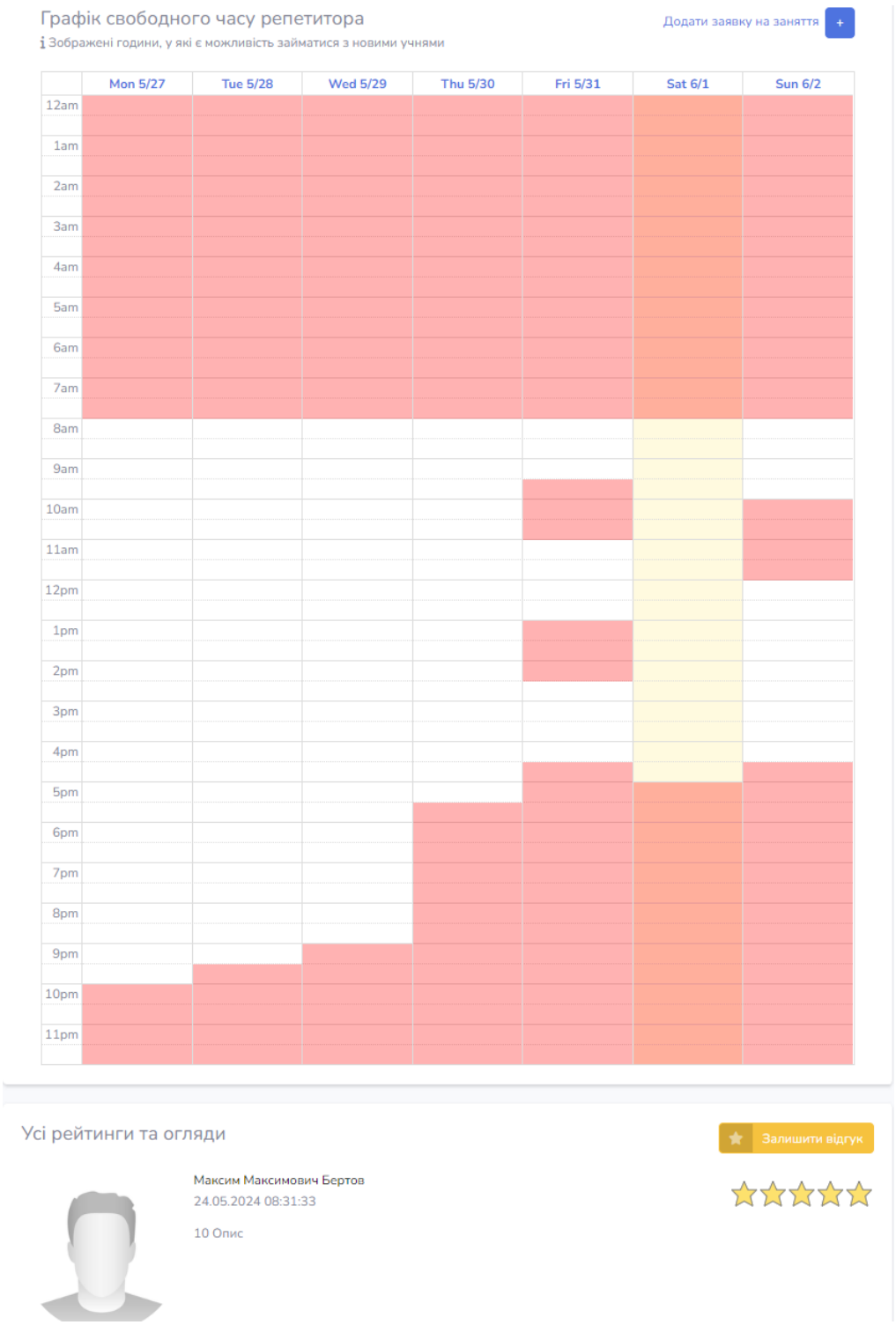


Рисунок 4.14 – Приклад відображення графіку свободного часу та відгуків у профілі репетитора

Висновки до розділу

Розділ зосереджувався на аналізі якості програмного забезпечення та описі процесів тестування. Було проаналізовано код за допомогою платформи Embold. Результати аналізу якості коду показали, що програма має стабільну архітектуру та відповідає вимогам щодо безпеки та ефективності.

Було проведено мануальне тестування, спрямоване на перевірку функціональності та стабільності системи. Усі ці тести були успішно завершені, що підтверджує надійність та працездатність програмного продукту. Далі було продемонстровано контрольний приклад зі створення, перегляду профілю та пошуку профілей репетиторів, який включав в себе ряд тестів, спрямованих на перевірку різних функцій системи. Результати цих тестів підтвердили ефективність та коректну роботу системи у реальних умовах використання.

На основі проведених досліджень можна зробити висновок, що розроблена система відповідає вимогам якості та функціональності. Вона має стабільну архітектуру, надійність та забезпечує користувачам потрібний функціонал для успішного виконання поставлених завдань.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для розгортання розробленого програмного забезпечення треба клонувати код програмних модулів з GitHub. Код усіх компонентів розробки розміщений у репозиторії, який доступний для вільного клонування. Щоб клонувати код потрібно використати команду git clone з URL репозиторію. Це надасть локальну копію коду програми на вашому комп'ютері.

5.2 Супровід програмного забезпечення

Інструкція користувача та системного адміністратора/програміста наведена в окремому документі.

Для забезпечення стабільної підтримки цього програмного забезпечення рекомендовано використовувати систему керування версіями Git та сервіс GitHub. Користувачі, які вже налаштували та розгорнули застосунок, можуть отримати останні оновлення, виконавши команду git pull у своєму локальному репозиторії. Це дозволить їм отримати найсвіжіші зміни в коді програмного забезпечення.

Висновки до розділу

В розділі було розглянуто ключові аспекти розгортання та супроводу програмного забезпечення. Процес розгортання починається з клонування коду з репозиторію на GitHub, що забезпечує легкий доступ до всіх програмних компонентів. Користувачі можуть швидко створити локальну копію коду на своєму комп'ютері, використовуючи команду git clone.

Супровід програмного забезпечення передбачає регулярне оновлення локальних репозиторіїв за допомогою команди git pull. Це дозволяє користувачам отримувати останні зміни та виправлення, забезпечуючи стабільну роботу програмного забезпечення. Використання Git та GitHub значно спрощує цей процес, надаючи інструменти для керування версіями та спільної роботи над кодом.

ВИСНОВКИ

Метою дипломного проєкту є підвищити ефективність пошуку та підбору розкладу занять з репетитором.

Під час передпроектного дослідження було проаналізовано аналоги вебзастосунків для пошуку репетиторів, що допомогло визначити основні напрямки розробки та розширити функціонал існуючих проектів.

Для побудови розгалуженої архітектури були використані шаблони проектування MVC та CQRS, які сприяють розділенню обов'язків та підвищенню гнучкості системи. Крім того, обрано чисту архітектуру завдяки її перевагам у розділенні відповідальностей, покращеній підтримці тестування, гнучкості та зручності обслуговування коду.

В якості середовища розробки обрано Visual Studio Community завдяки повній підтримці мові програмування C# та великій кількість плагінів і розширень, які дозволяють додати нові функції або інтегруватися з іншими інструментами та сервісами.

У якості БД використано MySQL через широку підтримку, продуктивність та маштабованість.

Особлива увага у проекті була приділена розширенню функціоналу, а саме можливості додавання завдань для учнів, додавання відповідей та їхній перегляд викладачем. Було розширено опції фільтрації для пошуку репетиторів.

Результати аналізу коду та тестування підтвердили високу надійність та точність роботи програмного застосунку, гарантуючи його адекватне функціонування під час реального використання.

Дана робота забезпечує у собі можливість виставлення завдань репетитором для учня та пошук викладачів в одному вебзастосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Аналіз ринку приватного репетиторства [Електронний ресурс] – Режим доступу до ресурсу: https://www.technavio.com/report/private-tutoring-market-industry-analysis.
2) Куллінан, Т. Останні тенденції в приватному та шкільному репетиторстві / Т. Куллінан, Р. Монтакуте. – 2023.
3) Документація мови C# [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/ru-ru/dotnet/csharp/.
4) Опис СУБД MySQL. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/.
5) Опис фреймворку Entity Framework Core. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/enus/ef/core/.
6) Чиста архітектура .NET Core [Електронний ресурс] – Режим доступу до ресурсу: https://www.tatvasoft.com/blog/clean-architecture-net-core/.
7) Вступ до ідентифікації на ASP.NET Core [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio.
8) Загальні відомості про ASP.NET Core [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0.
9) Embold. Платформа статичного аналізу коду. [Електронний ресурс] – Режим доступу до ресурсу: https://embold.io

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK
by Turnitin

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016331307

Дата перевірки:
07.06.2024 11:03:39 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.06.2024 13:18:10 EEST

ID користувача:
76913

Назва документа: **IT-02_Скачкова_ПЗ_070624**

Кількість сторінок: **99** Кількість слів: **14396** Кількість символів: **114287** Розмір файлу: **5.52 MB** ID файлу: **1016130908**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

12.6%
Схожість

Найбільша схожість: **4.24%** з джерелом з Бібліотеки (ID файлу: **1016116305**)

2.01% Джерела з Інтернету	264	Сторінка 101
12.6% Джерела з Бібліотеки	474	Сторінка 103

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0.66%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **8 слів** та **0%**)

0% Вилучення з Інтернету	90	Сторінка 104
0.66% Вилученого тексту з Бібліотеки	64	Сторінка 104

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	20 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ПОШУКУ РЕПЕТИТОРІВ

Текст програми

КПІ.ІТ-0221.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Анастасія СКАЧКОВА

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
https://github.com/Nastkav/TutorSearch
Нижче можна переглянути основні файли проекту:
Файл Program.cs

```
using System.Reflection;
using System.Text.Json.Serialization;
using Domain;
using Domain.Helpers;
using Infra;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using Infra.Storage;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using NuGet.Protocol;
using Web;
using Web.Middlewares;


var builder = WebApplication.CreateBuilder(args);
var services = builder.Services;
//Config
//Infra
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
if (connectionString == null)
    throw new InvalidOperationException("Connection string 'DefaultConnection' not found.");

services.AddDbContext<AppDbContext>(options => options.UseMySQL(connectionString));
services.AddIdentity<UserModel, IdentityRole<int>>(o => o.SignIn.RequireConfirmedEmail = false)
    .AddRoles<IdentityRole<int>>()
    .AddEntityFrameworkStores<AppDbContext>()
    .AddSignInManager()
    .AddDefaultTokenProviders()
    .AddDefaultUI();

services.Configure<SecurityStampValidatorOptions>(o => o.ValidationInterval = TimeSpan.FromSeconds(10));

services.AddTransient<IStorageRepository, StorageRepository>();

services.AddAuthentication()
    .AddCookie(options =>
    {
        options.LoginPath = "/Account/Login";
        options.AccessDeniedPath = "/Account/Forbidden/";
    });

//Domain
services.AddMediatR(cfg => cfg.RegisterServicesFromAssembly(typeof(BaseMediatrHandler<,>).Assembly));
services.AddAutoMapper(typeof(DomainMappingProfile));

//Web
services.AddEndpointsApiExplorer();
services.AddSwaggerGen();
services.AddProblemDetails();
services.AddControllersWithViews()
    .AddJsonOptions(options => //Опція щоб віддавати enum списки не числами, а рядковими значеннями
        options.JsonSerializerOptions.Converters.Add(new JsonStringEnumConverter()));
services.AddRazorPages();


//Запуск програми
var app = builder.Build();
```

```
app.UseExceptionHandler(); // Should be always in first place
//Оновлення бази даних
app.UseMigrationsEndPoint();

//Swagger
app.UseSwagger();
app.UseSwaggerUI();

//wwwroot
app.UseStaticFiles();
//Додавання маршрутизації
app.UseRouting();

//User Auth
app.UseAuthentication();
app.UseAuthorization();

//Routing
app.MapControllerRoute("default", "{controller=Profile}/{action=Index}/{id?}");
app.MapRazorPages(); //UserIdentity // Потреба в ідентифікації користувача

//Deleted
app.UseMiddleware<RemovedRoleMiddleware>();


app.Run();
```

Файл AddOrUpdateReviewCommand.cs

```
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.EntityFrameworkCore;

namespace Domain.Commands;

public class AddOrUpdateReviewCommand : IRequest<Review>
{
    public Review Review { get; set; } = null!;

    public class AddOrUpdateReviewCommandHandler : BaseMediatrHandler<AddOrUpdateReviewCommand, Review>
    {
        public AddOrUpdateReviewCommandHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }

        public override async Task<Review> Handle(AddOrUpdateReviewCommand r, CancellationToken token)
        {
            //Перевірка логіки
            if (r.Review is null)
                throw new CommandParameterException("Review is null");

            if (r.Review.AuthorId == r.Review.TutorId)
                throw new CommandParameterException("Не можна залишити відгук самому собі");

            var tutorExist = await DatabaseContext.Tutors.AnyAsync(x => x.Id == r.Review.TutorId);
            if (!tutorExist)
                throw new UserNotFoundException("Вчителя з таким профілем не знайдено");

            var dbReview = await DatabaseContext.Reviews
                .Include(x => x.Author)
                .Include(x => x.Tutor)
                .ThenInclude(x => x.User)
                .FirstOrDefaultAsync(x =>
```

```
                x.AuthorId == r.Review.AuthorId && x.TutorId == r.Review.TutorId);

            if (dbReview == null) //Доданя нового відгука
            {
                var countLessons = DatabaseContext.Lessons.Count(l =>
                    l.TutorId == r.Review.TutorId &&
                    l.Students.Any(s => s.Id == r.Review.AuthorId));
                if (countLessons == 0)
                    throw new LessonException("У вас не було жодного уроку з цим викладачем");

                dbReview = Mapper.Map<ReviewModel>(r.Review);
                await DatabaseContext.AddAsync(dbReview);
                DatabaseContext.Entry(dbReview).Reference(x => x.Author).Load();
                DatabaseContext.Entry(dbReview).Reference(x => x.Tutor).Load();
                DatabaseContext.Entry(dbReview.Tutor).Reference(x => x.User).Load();
            }
            else //Оновленя існуючого відгука
            {
                dbReview.Rating = r.Review.Rating;
                dbReview.Description = r.Review.Description;
                DatabaseContext.Update(dbReview);
            }

            //Збереження у базу даних
            await DatabaseContext.SaveChangesAsync();
            //Видача оновленого запису
            var review = Mapper.Map<Review>(dbReview);

            return review;
        }
    }
}
```

Файл CheckFavoriteCommand.cs

```
using System.ComponentModel.DataAnnotations;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using AutoMapper.Internal;
using Domain.Helpers;
using Microsoft.EntityFrameworkCore;

namespace Domain.Commands;

public class CheckFavoriteCommand : IRequest<bool>
{
    [Range(1, int.MaxValue)] public int UserId { get; set; }
    [Range(1, int.MaxValue)] public int TutorId { get; set; }
    public bool Status { get; set; }

    public class CheckFavoriteCommandHandler : BaseMediatrHandler<CheckFavoriteCommand, bool>
    {
        public CheckFavoriteCommandHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }


        public override async Task<bool> Handle(CheckFavoriteCommand r, CancellationToken token)
        {
            var dbUser = await DatabaseContext.Users
                .Include(x => x.FavoriteTutors)
                .FirstOrDefaultAsync(x => x.Id == r.UserId);
            if (dbUser == null)
                throw new UserNotFoundException("Неправильний ідентифікатор користувача");
```

```
            var dbTutor = await DatabaseContext.Tutors.FirstOrDefaultAsync(x => x.Id == r.TutorId);
            if (dbTutor == null)
                throw new UserNotFoundException("Неправильний ідентифікатор вчителя");

            var dbFav = dbUser.FavoriteTutors.FirstOrDefault(x => x.Id == r.TutorId);
            if (dbFav == null && r.Status)
                dbUser.FavoriteTutors.Add(dbTutor);
            else if (dbFav != null && !r.Status)
                dbUser.FavoriteTutors.Remove(dbFav);


            await DatabaseContext.SaveChangesAsync();
            return r.Status;
        }
    }
}
```

Файл CreateAssignmentCommand.cs

```
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Commands;

public class CreateAssignmentCommand : IRequest<int>
{
    public int? CreatedId { get; set; }
    public Assignment Assignment { get; set; } = null!;

    public class CreateAssignmentCommandHandler : BaseMediatrHandler<CreateAssignmentCommand, int>
    {
        public CreateAssignmentCommandHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }

        public override async Task<int> Handle(CreateAssignmentCommand r, CancellationToken token)
        {
            if (r.CreatedId != r.Assignment.TutorId)
                throw new CommandParameterException("Лише репетитор може додати завдання");
            if (!DatabaseContext.Tutors.Any(x => x.Id == r.Assignment.TutorId))
                throw new UserNotFoundException("Репетитор вказан невірно");

            var dbSubject = await DatabaseContext.Subjects
                .FirstOrDefaultAsync(x => x.Id == r.Assignment.SubjectId || x.Name == r.Assignment.SubjectName);
            if (dbSubject == null)
                throw new SubjectNotFoundException(
                    $"Предмету '{r.Assignment.SubjectId}:{r.Assignment.SubjectName}' не знайдено.");

            var newAssignment = Mapper.Map<AssignmentModel>(r.Assignment);
            newAssignment.Subject = dbSubject;

            //Додання рішень
            foreach (var studentId in r.Assignment.StudentsIds)
                newAssignment.Solutions.Add(new SolutionModel()
                {
                    Assignment = newAssignment,
                    StudentId = studentId,
                    Status = SolutionStatus.Todo
                });
```

```
            await DatabaseContext.Assignments.AddAsync(newAssignment);
            await DatabaseContext.SaveChangesAsync();
            return newAssignment.Id;
        }
    }
}
```

Файл CreateLessonCommand.cs

```
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Commands;

public class CreateLessonCommand : IRequest<int>
{
    public int CreatedId { get; set; }
    public Lesson Lesson { get; set; } = null!;

    public class CreateRequestCommandHandler : BaseMediatrHandler<CreateLessonCommand, int>
    {
        public CreateRequestCommandHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }

        public override async Task<int> Handle(CreateLessonCommand r, CancellationToken token)
        {
            if (r.CreatedId != r.Lesson.TutorId)
                throw new CommandParameterException("Лише репетитор може створити зустріч");
            if (!DatabaseContext.Tutors.Any(x => x.Id == r.Lesson.TutorId))
                throw new UserNotFoundException("Репетитор вказан невірно");

            var dbSubject = await DatabaseContext.Subjects
                .FirstOrDefaultAsync(x => x.Id == r.Lesson.SubjectId || x.Name == r.Lesson.SubjectName);
            if (dbSubject == null)
                throw new SubjectNotFoundException(
                    $"Предмету '{r.Lesson.SubjectId}:{r.Lesson.SubjectName}' не знайдено.");

            // // remove seconds
            r.Lesson.From = r.Lesson.From.Date.AddHours(r.Lesson.From.Hour).AddMinutes(r.Lesson.From.Minute);
            r.Lesson.To = r.Lesson.To.Date.AddHours(r.Lesson.To.Hour).AddMinutes(r.Lesson.To.Minute);
            //map
            var newLesson = Mapper.Map<LessonModel>(r.Lesson);
            newLesson.Subject = dbSubject;

            //Додання учнів
            newLesson.Students = DatabaseContext.Users.Where(x => r.Lesson.StudentsIds.Contains(x.Id)).ToList();

            //Перевірка перетинання часу
            //aF > bT and bF > aT
            if (await DatabaseContext.Lessons.CountAsync(x => x.From > r.Lesson.To && r.Lesson.From > x.To) > 0)
                throw new LessonException("У вас вже є зустріч у цей час");

            await DatabaseContext.Lessons.AddAsync(newLesson);
            await DatabaseContext.SaveChangesAsync();
            return newLesson.Id;
        }
    }
```

```
}
```

Файл CreateRequestCommand.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Commands;

public class CreateRequestCommand : IRequest<int>
{
    public int CreatedId { get; set; }
    public int TutorId { get; set; }
    [DisplayName("Початок")] public DateTime? From { get; set; }
    [DisplayName("Кінець")] public DateTime? To { get; set; }


    [Range(1, int.MaxValue, ErrorMessage = "Оберіть предмет")]
    public int? SubjectId { get; set; }

    public List<SelectListItem> Subjects { get; set; } = [];

    [MaxLength(300)]
    [MinLength(0)]
    [DisplayName("Коментар учня")]
    public string Comment { get; set; } = string.Empty;


    public class CreateRequestCommandHandler : BaseMediatrHandler<CreateRequestCommand, int>
    {
        public CreateRequestCommandHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }

        public override async Task<int> Handle(CreateRequestCommand r, CancellationToken token)
        {
            if (r.CreatedId == r.TutorId)
                throw new AccessDeniedException("Користувач може надіслати запрошення лише іншому репетитору");
            if (!DatabaseContext.Tutors.Any(x => x.Id == r.TutorId))
                throw new UserNotFoundException("Репетитор вказан невірно");
            if (!DatabaseContext.Users.Any(x => x.Id == r.CreatedId))
                throw new UserNotFoundException("Ученик вказан невірно");

            var dbSubject = await DatabaseContext.Subjects.FirstOrDefaultAsync(x => x.Id == r.SubjectId);
            if (dbSubject == null)
                throw new Exception($"Предмету '{r.SubjectId}' не знайдено.");

            var newRequest = Mapper.Map<RequestModel>(r);
            newRequest.Subject = dbSubject;
            newRequest.Status = LessonRequestStatus.New;

            //Перевірка існування схожого запиту до цього викладача
            var requestExist = DatabaseContext.Requests.Any(x =>
                x.CreatedId == newRequest.CreatedId &&
                x.Status == newRequest.Status &&
                x.TutorId == newRequest.TutorId &&
                x.Subject.Id == newRequest.Subject.Id
            );
```

```
        if (requestExist)
            throw new RequestException("Ви вже маєте активний запит на курс для цього викладача");

        //Перевірка перетинання часу
        //aF > bT and bF > aT
        if (await DatabaseContext.Lessons.CountAsync(x => x.From > newRequest.To && newRequest.From > x.To) >
0)
            throw new RequestException("Додавання неможливе, час перетинається");

        await DatabaseContext.Requests.AddAsync(newRequest);
        await DatabaseContext.SaveChangesAsync();
        return newRequest.Id;
    }
  }
}
```

Файл CreateSessionCommand.cs

```
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.Extensions.Logging;

namespace Domain.Commands;

public class CreateSessionCommand : IRequest<int>
{
  public int CreatedBy { get; set; }

  public TimeType Type { get; set; }
  public string Title { get; set; } = "";
  public DateTimeOffset From { get; set; }
  public DateTimeOffset To { get; set; }


  public class CreateSessionCommandHandler : BaseMediatrHandler<CreateSessionCommand, int>
  {
    public CreateSessionCommandHandler(AppDbContext dbContext, IMapper mapper)
      : base(dbContext, mapper)
    {
    }

    public override async Task<int> Handle(CreateSessionCommand r, CancellationToken token)
    {
      //On day event
      if (r.From.Date != r.To.Date)
        throw new TimeRangeException("Подія має бути протягом дня.");
      else if (r.From > r.To || r.From == r.To)
        throw new TimeRangeException("Обрано невірний час");

      var weekday = (int)r.From.DayOfWeek;
      var start = TimeOnly.FromDateTime(r.From.DateTime);
      var end = TimeOnly.FromDateTime(r.To.DateTime);

      switch (r.Type)
      {
        case TimeType.Available: //Встановлення робочого часу репетитора у кабінеті
          var dbTimes = DatabaseContext.AvailableTimes
            .Where(x => x.ProfileId == r.CreatedBy && x.DayOfWeek == weekday)
            .OrderBy(x => x.StartTime)
            .ToList();

          //Перевірка перетинання часу
          foreach (var timeRange in dbTimes)
            if (timeRange.EndTime > start && timeRange.StartTime < end)
```

```
                throw new TimeRangeException("Додавання неможливе, час перетинається");

            var availableTime = new AvailableTimeModel()
            {
                DayOfWeek = weekday,
                StartTime = TimeOnly.FromDateTime(r.From.DateTime),
                EndTime = TimeOnly.FromDateTime(r.To.DateTime),
                ProfileId = r.CreatedBy,
                CreatedId = r.CreatedBy
            };
            DatabaseContext.AvailableTimes.Add(availableTime);
            await DatabaseContext.SaveChangesAsync();
            return availableTime.Id;
        default:
            throw new ArgumentOutOfRangeException("Неможливо додати подію типу: " + r.Type.ToString());
    }
  }
 }
}
```

Файл DeleteAssignmentCommand.cs

```
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Commands;

public class DeleteAssignmentCommand : IRequest<bool>
{
  public int AssignmentId { get; set; }
  public int TutorId { get; set; }

  public class DeleteAssignmentCommandHandler : BaseMediatrHandler<DeleteAssignmentCommand, bool>
  {
    public override async Task<bool> Handle(DeleteAssignmentCommand r, CancellationToken token)
    {
      var assignment = await DatabaseContext.Assignments
        .FirstOrDefaultAsync(x => x.TutorId == r.TutorId && x.Id == r.AssignmentId);
      if (assignment == null)
        return false;

      DatabaseContext.Remove(assignment);
      await DatabaseContext.SaveChangesAsync();
      return true;
    }

    public DeleteAssignmentCommandHandler(AppDbContext dbContext, IMapper mapper)
      : base(dbContext, mapper)
    {
    }
  }
}
```

Файл DeleteFileCommand.cs

```
using System.Configuration;
using System.Data;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra;
using Infra.Storage;
```

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;

namespace Domain.Commands;

public class DeleteFileCommand : IRequest<bool>
{
   public string FileId { get; set; } = string.Empty;
   public int? UserId { get; set; }

   public class DeleteFileCommandHandler : BaseMediatrHandler<DeleteFileCommand, bool>
   {
      private readonly IStorageRepository _storageRepo;

      public DeleteFileCommandHandler(AppDbContext dbContext, IMapper mapper, IStorageRepository storageRepo) :
         base(dbContext, mapper) => _storageRepo = storageRepo;


      public override async Task<bool> Handle(DeleteFileCommand r, CancellationToken token)
      {
         var dbFile = await DatabaseContext.Files.FirstOrDefaultAsync(x => x.Id.ToString() == r.FileId);
         if (dbFile == null)
            throw new CommandParameterException("Файл не знайдено у базі даних");
         if (r.UserId == 0 || dbFile.OwnerId != r.UserId)
            throw new CommandParameterException("Тільки власник може видалити файл");

         _storageRepo.RemoveFileIfExist(dbFile.ServerName);

         //Видалення файлу з БД
         DatabaseContext.Remove(dbFile);
         await DatabaseContext.SaveChangesAsync();
         return true;
      }
   }
}
```

Файл DeleteLessonCommand.cs

```
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Commands;

public class DeleteLessonCommand : IRequest<bool>
{
   public int LessonId { get; set; }
   public int TutorId { get; set; }

   public class DeleteLessonCommandHandler : BaseMediatrHandler<DeleteLessonCommand, bool>
   {
      public override async Task<bool> Handle(DeleteLessonCommand r, CancellationToken token)
      {
         var lesson = await DatabaseContext.Lessons
            .FirstOrDefaultAsync(x => x.TutorId == r.TutorId && x.Id == r.LessonId);
         if (lesson == null)
            return false;

         DatabaseContext.Remove(lesson);
         await DatabaseContext.SaveChangesAsync();
         return true;
      }

      public DeleteLessonCommandHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
```

```
        {
        }
    }
}
```

Файл DeleteSessionCommand.cs

```
using System.Data;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Domain.Models;

namespace Domain.Commands;

public class DeleteSessionCommand : IRequest<bool>
{
    public TimeType Type { get; set; }
    public int UpdatedBy { get; set; }
    public int EventId { get; set; }

    public class DeleteSessionCommandHandler : BaseMediatrHandler<DeleteSessionCommand, bool>
    {
        public override async Task<bool> Handle(DeleteSessionCommand r, CancellationToken token)
        {
            if (r.Type == TimeType.Available)
            {
                var lesson = DatabaseContext.AvailableTimes
                    .FirstOrDefault(x => x.Id == r.EventId && x.CreatedId == r.UpdatedBy);
                if (lesson == null)
                    throw new ArgumentNullException("Помилка в номері події або користувач не має доступу");
                DatabaseContext.Remove(lesson);
            }
            else
            {
                var lesson = DatabaseContext.Lessons.FirstOrDefault(x => x.Id == r.EventId && x.TutorId == r.UpdatedBy);
                if (lesson == null)
                    throw new ArgumentNullException("Помилка в номері події");
                DatabaseContext.Remove(lesson);
            }

            await DatabaseContext.SaveChangesAsync();
            return true;
        }

        public DeleteSessionCommandHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл SaveAvatarCommand.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Net.Mime;
using System.Text;
using AutoMapper;
using Domain.Helpers;
using Infra;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using Infra.Storage;
using MediatR;
using Microsoft.AspNetCore.Hosting;
```

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Http.Internal;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.Formats.Png;
using SixLabors.ImageSharp.Processing;

namespace Domain.Commands;

public class SaveAvatarCommand : IRequest<bool>
{
   public int UserId { get; set; }

   public byte[] ImageBytes { get; set; } = [];

   public class SaveAvatarCommandHandler : IRequestHandler<SaveAvatarCommand, bool>
   {
      private readonly IStorageRepository _storage;

      public SaveAvatarCommandHandler(IStorageRepository storage) =>
         _storage = storage;

      public Task<bool> Handle(SaveAvatarCommand r, CancellationToken token)
      {
         if (r.ImageBytes.Length == 0)
            throw new CommandParameterException("Файл не може бути пустим");

         _storage.SaveAvatar(r.UserId.ToString(), r.ImageBytes);
         return Task.FromResult(true);
      }
   }
}
```

Файл UpdateAssignmentCommand.cs

```
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Commands;

public class UpdateAssignmentCommand : IRequest<Assignment>
{
   public int UpdatedBy { get; set; }
   public int AssignmentId { get; set; }
   public string? Title { get; set; } = null;
   public string? Description { get; set; } = null;
   public DateTime? Deadline { get; set; }
   public int? SubjectId { get; set; }
   public List<int> StudentIds { get; set; } = [];


   public class UpdateAssignmentCommandHandler : BaseMediatrHandler<UpdateAssignmentCommand, Assignment>
   {
      public override async Task<Assignment> Handle(UpdateAssignmentCommand r, CancellationToken token)
      {
         var dbAssignment = DatabaseContext.Assignments
            .Include(x => x.Solutions)
            .FirstOrDefault(x => x.Id == r.AssignmentId);
         if (dbAssignment == null)
```

```
            throw new AssignmentException("Задачу не знайдено");
        if (r.UpdatedBy == 0)
            throw new UserNotFoundException("Користувача не вказано");
        if (r.UpdatedBy != dbAssignment.TutorId)
            throw new CommandParameterException("Редагувати завдання може лише вчитель");
        if (r.Deadline.HasValue && r.Deadline < DateTime.Today)
            throw new TimeRangeException("Термін має бути у майбутньому.", r.Deadline.Value);

        if (r.Title != null) dbAssignment.Title = r.Title;
        if (r.Description != null) dbAssignment.Description = r.Description;
        if (r.Deadline != null) dbAssignment.Deadline = r.Deadline.Value;
        if (r.SubjectId != null) dbAssignment.SubjectId = r.SubjectId.Value;

        //Видалити рішення які ще не виконані та їх немає у списку
        foreach (var s in dbAssignment.Solutions)
            if (!r.StudentIds.Contains(s.StudentId) && s.Status == SolutionStatus.Todo)
                DatabaseContext.Solutions.Remove(s);

        var studentsIds = dbAssignment.Solutions.Select(x => x.StudentId).ToList();
        //Видалення існуючих рішень студентів зі списку вставки
        foreach (var studId in studentsIds)
            r.StudentIds.Remove(studId);

        //Додати нові рішення рішення
        foreach (var studentId in r.StudentIds)
            dbAssignment.Solutions.Add(new SolutionModel()
            {
                AssignmentId = dbAssignment.Id,
                StudentId = studentId,
                Status = SolutionStatus.Todo
            });

        //Зберегти
        DatabaseContext.Assignments.Update(dbAssignment);
        await DatabaseContext.SaveChangesAsync();
        //Повернути оновлене завдання
        var task = Mapper.Map<Assignment>(dbAssignment);
        return task;
    }

    public UpdateAssignmentCommandHandler(AppDbContext dbContext, IMapper mapper)
        : base(dbContext, mapper)
    {
    }
  }
}
```

Файл UpdateRequestCommand.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Commands;

public class UpdateRequestCommand : IRequest<int>
{
  [DisplayName("Запит №")] public int Id { get; set; }
  [StringLength(150)] public string? Subject { get; set; }
  public int TutorId { get; set; }
```

```
    [DisplayName("Початок")] public DateTime? From { get; set; }
    [DisplayName("Кінець")] public DateTime? To { get; set; }

    [DisplayName("Коментар вчителя")]
    [StringLength(254, ErrorMessage = "{0} має містити не більше {1} символів.", MinimumLength = 0)]
    public string TutorComment { get; set; } = string.Empty;

    [DisplayName("Статус")] public LessonRequestStatus Status { get; set; } = LessonRequestStatus.New;
    public int UpdatedBy { get; set; }

    public class UpdateRequestCommandHandler : BaseMediatrHandler<UpdateRequestCommand, int>
    {
        public UpdateRequestCommandHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }

        public override async Task<int> Handle(UpdateRequestCommand r, CancellationToken token)
        {
            if (r.UpdatedBy != r.TutorId)
                throw new RequestException("Тільки репетитор може оновлювати запити");

            var dbRequest = DatabaseContext.Requests
                .Include(x => x.Subject)
                .Include(x => x.Created)
                .FirstOrDefault(x => x.Id == r.Id && x.TutorId == r.TutorId);
            if (dbRequest is null)
                throw new RequestException("Потрібний запит не знайдено");

            if (dbRequest.Status != LessonRequestStatus.New)
                throw new RequestException("Заявка вже закрита");

            Mapper.Map(r, dbRequest);
            if (r.Subject != null)
                dbRequest.Subject = DatabaseContext.Subjects
                    .First(x => x.Name == r.Subject);
            DatabaseContext.Requests.Update(dbRequest);

            if (dbRequest.Status == LessonRequestStatus.Approved) // Створення нового уроку
            {
                var lesson = Mapper.Map<LessonModel>(dbRequest);
                lesson.Students.Add(dbRequest.Created);
                await DatabaseContext.Lessons.AddAsync(lesson);
            }

            await DatabaseContext.SaveChangesAsync();
            return dbRequest.Id;
        }
    }
}
```

Файл UpdateSolutionCommand.cs

```
using System.ComponentModel.DataAnnotations;
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Commands;

public class UpdateSolutionCommand : IRequest<int>
```

```
{
    public int UpdatedBy { get; set; }
    public int SolutionId { get; set; }
    public SolutionStatus? Status { get; set; }
    public string? Answer { get; set; } = string.Empty;
    public string? TutorComment { get; set; } = string.Empty;

    public class UpdateSolutionCommandHandler : BaseMediatrHandler<UpdateSolutionCommand, int>
    {
        public override async Task<int> Handle(UpdateSolutionCommand r, CancellationToken token)
        {
            var dbSolution = DatabaseContext.Solutions
                .Include(x => x.Assignment)
                .FirstOrDefault(x => x.Id == r.SolutionId);
            if (dbSolution == null)
                throw new SolutionException("Задачу не знайдено");
            if (dbSolution.Status == SolutionStatus.Completed)
                throw new SolutionException("Оновленя виконаної задачі неможливо");
            var isStudent = r.UpdatedBy == dbSolution.StudentId;
            var isTutor = r.UpdatedBy == dbSolution.Assignment.TutorId;
            if (!isStudent && !isTutor)
                throw new AccessDeniedException("Редагувати завдання може лише вчитель або учень");

            //Status
            if (r.Status != null && dbSolution.Status != r.Status)
            {
                if (isStudent && r.Status.Value == SolutionStatus.Completed)
                    throw new AccessDeniedException("Учень не може відзначити завдання як виконане");
                dbSolution.Status = r.Status.Value;
            }

            //Answer
            if (r.Answer != null && dbSolution.Answer != r.Answer)
                if (isStudent)
                    dbSolution.Answer = r.Answer;
                else
                    throw new AccessDeniedException("Вчитель не може написати відповідь");


            //TutorComment
            if (r.TutorComment != null && dbSolution.TutorComment != r.TutorComment)
                if (isTutor)
                    dbSolution.TutorComment = r.TutorComment;
                else
                    throw new AccessDeniedException("Учень не може написати коментар");


            //Зберегти
            DatabaseContext.Solutions.Update(dbSolution);
            await DatabaseContext.SaveChangesAsync();
            return dbSolution.Id;
        }

        public UpdateSolutionCommandHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл UpdateTutorCommand.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Diagnostics.CodeAnalysis;
using AutoMapper;
using Domain.Helpers;
```

```
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.Extensions.Logging;
using Domain.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Commands;

public class UpdateTutorCommand : IRequest<bool>
{
   public required Tutor Profile { get; set; }

   public class EditTutorProfileCommandHandler : BaseMediatrHandler<UpdateTutorCommand, bool>
   {
      public EditTutorProfileCommandHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
      {
      }

      public override async Task<bool> Handle(UpdateTutorCommand r, CancellationToken token)
      {
         //Check exist
         if (!DatabaseContext.Users.Any(x => x.Id == r.Profile.Id && x.ProfileEnabled))
            throw new UserNotFoundException("Ідентіфікатор вчителя не знайдено.");

         //Select from database
         var dbProfile = await DatabaseContext.Tutors
                        .Include(x => x.About)
                        .Include(x => x.Subjects)
                        .FirstOrDefaultAsync(x => x.Id == r.Profile.Id)
                     ?? new TutorModel();
         var newObject = dbProfile.Id == 0;

         //Mapping
         Mapper.Map(r.Profile, dbProfile);

         //Subjects
         dbProfile.Subjects = await DatabaseContext.Subjects
            .Where(x => r.Profile.SubjectIds.Contains(x.Id))
            .ToListAsync();

         if (newObject)
            DatabaseContext.Add(dbProfile);
         else
            DatabaseContext.Update(dbProfile);

         await DatabaseContext.SaveChangesAsync(token);
         return true;
      }
   }
}
```

Файл UpdateUserCommand.cs

```
using System.Web;
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Commands;
```

```
public class UpdateUserCommand : IRequest<bool>
{
   public required User Profile { get; set; }

   public class UpdateUserCommandHandler : BaseMediatrHandler<UpdateUserCommand, bool>
   {
      public UpdateUserCommandHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
      {
      }

      public override async Task<bool> Handle(UpdateUserCommand r, CancellationToken token)
      {
         var dbUser = await DatabaseContext.Users.FirstOrDefaultAsync(x => x.Id == r.Profile.Id);
         //Check exist
         if (dbUser == null || dbUser.Id == 0 || !DatabaseContext.Users.Any(x => x.Id == dbUser.Id))
            throw new UserNotFoundException("Ідентіфікатор користувача не знайдено.");
         Mapper.Map(r.Profile, dbUser);
         DatabaseContext.Users.Update(dbUser);
         await DatabaseContext.SaveChangesAsync(token);
         return true;
      }
   }
}
```

Файл UploadFileCommand.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Configuration;
using System.Drawing;
using System.Drawing.Imaging;
using System.Net.Mime;
using AutoMapper;
using Domain.Helpers;
using Infra;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using Infra.Storage;
using MediatR;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;
using MySql.Data.MySqlClient;
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.Formats.Png;
using SixLabors.ImageSharp.Processing;

namespace Domain.Commands;

public class UploadFileCommand : IRequest<bool>
{
   public int? UserId { get; set; }
   public int? AssignmentId { get; set; }
   public int? SolutionId { get; set; }
   public byte[] FileBytes { get; set; } = [];
   public string FileName { get; set; } = string.Empty;

   public class UploadFileCommandHandler : BaseMediatrHandler<UploadFileCommand, bool>
   {
      private readonly IStorageRepository _storageRepo;

      public UploadFileCommandHandler(AppDbContext dbContext, IMapper mapper, IStorageRepository storageRepo) :
```

```
        base(dbContext, mapper) => _storageRepo = storageRepo;

    //https://blog.elmah.io/upload-and-resize-an-image-with-asp-net-core-and-imagesharp/
    public override async Task<bool> Handle(UploadFileCommand r, CancellationToken token)
    {
        //Перевірки
        if (!r.UserId.HasValue || r.UserId == 0)
            throw new UserNotFoundException("Користувач вказаний невірно");
        if (r.FileBytes.Length == 0)
            throw new CommandParameterException("Передача файлу обов'язкова");
        if (r.FileName == string.Empty)
            throw new CommandParameterException("Файл повинен мати назву");

        var dbAssignment = await DatabaseContext.Assignments
            .FirstOrDefaultAsync(x => x.Id == r.AssignmentId && x.TutorId == r.UserId);
        var dbSolution = await DatabaseContext.Solutions
            .Include(x => x.Assignment)
            .FirstOrDefaultAsync(x =>
                (x.Id == r.SolutionId && x.StudentId == r.UserId) || x.Assignment.TutorId == r.UserId);

        if (r.AssignmentId != null && dbAssignment == null)
            throw new AssignmentException("Невірно вказано номер завдання");
        if (r.SolutionId != null && dbSolution == null)
            throw new SolutionException("Невірно вказано номер рішення");

        var newFile = _storageRepo.SaveFile(Path.GetExtension(r.FileName), r.FileBytes);
        //Реєстрація нового рядка в БД
        var dbFile = new UserFileModel()
        {
            Id = newFile.Guid,
            FileName = r.FileName,
            ServerName = newFile.Filename,
            OwnerId = r.UserId.Value
        };

        //Посилання на файли із завданнями та рішеннями
        if (r.AssignmentId.HasValue && dbAssignment != null)
            dbFile.Assignments.Add(dbAssignment);
        if (r.SolutionId.HasValue && dbSolution != null)
            dbFile.Solutions.Add(dbSolution);

        //Зберегти зміни в новому рядку в базі даних
        DatabaseContext.Files.Add(dbFile);
        await DatabaseContext.SaveChangesAsync();
        return true;
    }
  }
}
```

Файл BaseMediatrHandler.cs

```
using AutoMapper;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;

namespace Domain.Helpers;

public abstract class BaseMediatrHandler<T, TU> : IRequestHandler<T, TU>
  where T : IRequest<TU>
{
  protected internal AppDbContext DatabaseContext { get; }
  protected internal IMapper Mapper { get; }

  public BaseMediatrHandler(AppDbContext dbContext, IMapper mapper)
  {
    DatabaseContext = dbContext;
```

```
        Mapper = mapper;
    }

    public abstract Task<TU> Handle(T r, CancellationToken token);
}
```

Файл DomainMappingProfile.cs

```
using AutoMapper;
using Domain.Commands;
using Domain.Models;
using Infra.DatabaseAdapter.Models;

namespace Domain.Helpers;

public class DomainMappingProfile : Profile
{
    public DomainMappingProfile()
    {
        //User & Tutor Profiles
        CreateMap<User, UserModel>()
            .ForMember(d => d.CityId, o => o.MapFrom(x => x.CityId == "0" ? null : x.CityId));

        CreateMap<UserModel, User>()
            .ForMember(d => d.FullName, o => o.MapFrom(x => x.FullName()))
            .ForMember(d => d.CityName, o => o.MapFrom(x => x.City == null ? "Не вказано" : x.City.Name));

        CreateMap<TutorModel, Tutor>()
            .ForMember(d => d.About, o => o.MapFrom(x => x.About.Content))
            .ForMember(d => d.ReviewCount, o => o.MapFrom(x => x.Reviews.Count))
            .ForMember(d => d.LessonCount, o => o.MapFrom(x => x.TeachingLessons.Count(l => l.To < DateTime.Now)))
            .ForMember(d => d.SubjectIds, o => o.MapFrom(x => x.Subjects.Select(s => s.Id).ToList()));
        //Нестабільна робота розрахунку середнього значення поля ReviewValue

        CreateMap<Tutor, TutorModel>()
            .ForPath(d => d.About.Content, o => o.MapFrom(x => x.About))
            .ForPath(d => d.About.Id, o => o.MapFrom(x => x.Id))
            .ForMember(d => d.Subjects, o => o.Ignore());

        //Files
        CreateMap<UserFileModel, UserFile>()
            .ForMember(d => d.OwnerName, o => o.MapFrom(x => x.Owner.Name));
        CreateMap<UserFile, UserFileModel>();


        //Requests
        CreateMap<LessonRequest, RequestModel>();
        CreateMap<RequestModel, LessonRequest>();
        CreateMap<CreateRequestCommand, RequestModel>();

        CreateMap<UpdateRequestCommand, RequestModel>()
            .ForMember(d => d.Subject, o => o.Ignore())
            .ForAllMembers(opts => { opts.Condition((_, _, srcMember) => srcMember != null); });

        CreateMap<RequestModel, LessonModel>()
            .ForMember(d => d.Id, o => o.Ignore())
            .ForMember(d => d.RequestId, o => o.MapFrom(x => x.Id))
            .ForMember(d => d.Comment, o => o.MapFrom(x => x.TutorComment));

        //Lessons
        CreateMap<LessonModel, Lesson>()
            .ForMember(d => d.StudentsIds, o => o.MapFrom(x => x.Students.Select(s => s.Id)))
            .ForMember(d => d.StudentNames, o => o.MapFrom(x =>
                string.Join(", ", x.Students.Select(s => s.FullName()))))
            .ForMember(d => d.TutorName, o => o.MapFrom(x => x.Tutor.User.FullName()));

        CreateMap<Lesson, LessonModel>()
```

```
            .ForMember(d => d.Students, o => o.Ignore())
            .ForMember(d => d.Tutor, o => o.Ignore());

        //Assignments
        CreateMap<Assignment, AssignmentModel>()
            // .ForMember(d => d.Deadline, o => o.MapFrom(x => x.Deadline.ToDateTime(TimeOnly.MinValue)))
            .ForMember(d => d.Files, opt => opt.MapFrom(s => s.FileNames));


        CreateMap<AssignmentModel, Assignment>()
            .ForMember(d => d.TutorName, o => o.MapFrom(x => x.Tutor.User.FullName()))
            .ForMember(d => d.SubjectName, o => o.MapFrom(x => x.Subject.Name))
            .ForMember(d => d.Deadline, o => o.MapFrom(x => x.Deadline))
            .ForMember(d => d.FileNames, opt => opt.MapFrom(s => s.Files))
            .ForMember(d => d.StudentsIds, o => o.MapFrom(x => x.Solutions.Select(s => s.StudentId)))
            .ForMember(d => d.StudentNames, o => o.MapFrom(x =>
                string.Join(", ", x.Solutions.Select(s => s.Student.FullName()))));

        //Solutions
        CreateMap<Solution, SolutionModel>()
            .ForMember(d => d.Files, opt => opt.MapFrom(s => s.SolutionFiles));

        CreateMap<SolutionModel, Solution>()
            .ForMember(d => d.StudentName, o => o.MapFrom(x => x.Student.FullName()))
            .ForMember(d => d.SolutionFiles, opt => opt.MapFrom(s => s.Files))
            .ForMember(d => d.TutorId, opt => opt.MapFrom(s => s.Assignment.Tutor.Id))
            .ForMember(d => d.TutorName, o => o.MapFrom(x => x.Assignment.Tutor.User.FullName()))
            .ForMember(d => d.AssignmentTitle, o => o.MapFrom(x => x.Assignment.Title))
            .ForMember(d => d.Description, o => o.MapFrom(x => x.Assignment.Description))
            .ForMember(d => d.SubjectName, o => o.MapFrom(x => x.Assignment.Subject.Name))
            .ForMember(d => d.AssignmentFiles, opt => opt.MapFrom(s => s.Assignment.Files));


        //Review and Feedbacks
        CreateMap<Review, ReviewModel>();
        CreateMap<ReviewModel, Review>()
            .ForMember(d => d.TutorName, o => o.MapFrom(x => x.Tutor.User.FullName()))
            .ForMember(d => d.AuthorName, o => o.MapFrom(x => x.Author.FullName()));
    }
}
```

Файл DomainMappingProfile.cs

```
using AutoMapper;
using Domain.Commands;
using Domain.Models;
using Infra.DatabaseAdapter.Models;

namespace Domain.Helpers;

public class DomainMappingProfile : Profile
{
    public DomainMappingProfile()
    {
        //User & Tutor Profiles
        CreateMap<User, UserModel>()
            .ForMember(d => d.CityId, o => o.MapFrom(x => x.CityId == "0" ? null : x.CityId));

        CreateMap<UserModel, User>()
            .ForMember(d => d.FullName, o => o.MapFrom(x => x.FullName()))
            .ForMember(d => d.CityName, o => o.MapFrom(x => x.City == null ? "Не вказано" : x.City.Name));

        CreateMap<TutorModel, Tutor>()
            .ForMember(d => d.About, o => o.MapFrom(x => x.About.Content))
            .ForMember(d => d.ReviewCount, o => o.MapFrom(x => x.Reviews.Count))
            .ForMember(d => d.LessonCount, o => o.MapFrom(x => x.TeachingLessons.Count(l => l.To < DateTime.Now)))
            .ForMember(d => d.SubjectIds, o => o.MapFrom(x => x.Subjects.Select(s => s.Id).ToList()));
```

```
//Нестабільна робота розрахунку середнього значення поля ReviewValue

CreateMap<Tutor, TutorModel>()
    .ForPath(d => d.About.Content, o => o.MapFrom(x => x.About))
    .ForPath(d => d.About.Id, o => o.MapFrom(x => x.Id))
    .ForMember(d => d.Subjects, o => o.Ignore());

//Files
CreateMap<UserFileModel, UserFile>()
    .ForMember(d => d.OwnerName, o => o.MapFrom(x => x.Owner.Name));
CreateMap<UserFile, UserFileModel>();


//Requests
CreateMap<LessonRequest, RequestModel>();
CreateMap<RequestModel, LessonRequest>();
CreateMap<CreateRequestCommand, RequestModel>();

CreateMap<UpdateRequestCommand, RequestModel>()
    .ForMember(d => d.Subject, o => o.Ignore())
    .ForAllMembers(opts => { opts.Condition((_, _, srcMember) => srcMember != null); });

CreateMap<RequestModel, LessonModel>()
    .ForMember(d => d.Id, o => o.Ignore())
    .ForMember(d => d.RequestId, o => o.MapFrom(x => x.Id))
    .ForMember(d => d.Comment, o => o.MapFrom(x => x.TutorComment));

//Lessons
CreateMap<LessonModel, Lesson>()
    .ForMember(d => d.StudentsIds, o => o.MapFrom(x => x.Students.Select(s => s.Id)))
    .ForMember(d => d.StudentNames, o => o.MapFrom(x =>
        string.Join(", ", x.Students.Select(s => s.FullName()))))
    .ForMember(d => d.TutorName, o => o.MapFrom(x => x.Tutor.User.FullName()));

CreateMap<Lesson, LessonModel>()
    .ForMember(d => d.Students, o => o.Ignore())
    .ForMember(d => d.Tutor, o => o.Ignore());

//Assignments
CreateMap<Assignment, AssignmentModel>()
    // .ForMember(d => d.Deadline, o => o.MapFrom(x => x.Deadline.ToDateTime(TimeOnly.MinValue)))
    .ForMember(d => d.Files, opt => opt.MapFrom(s => s.FileNames));


CreateMap<AssignmentModel, Assignment>()
    .ForMember(d => d.TutorName, o => o.MapFrom(x => x.Tutor.User.FullName()))
    .ForMember(d => d.SubjectName, o => o.MapFrom(x => x.Subject.Name))
    .ForMember(d => d.Deadline, o => o.MapFrom(x => x.Deadline))
    .ForMember(d => d.FileNames, opt => opt.MapFrom(s => s.Files))
    .ForMember(d => d.StudentsIds, o => o.MapFrom(x => x.Solutions.Select(s => s.StudentId)))
    .ForMember(d => d.StudentNames, o => o.MapFrom(x =>
        string.Join(", ", x.Solutions.Select(s => s.Student.FullName()))));

//Solutions
CreateMap<Solution, SolutionModel>()
    .ForMember(d => d.Files, opt => opt.MapFrom(s => s.SolutionFiles));

CreateMap<SolutionModel, Solution>()
    .ForMember(d => d.StudentName, o => o.MapFrom(x => x.Student.FullName()))
    .ForMember(d => d.SolutionFiles, opt => opt.MapFrom(s => s.Files))
    .ForMember(d => d.TutorId, opt => opt.MapFrom(s => s.Assignment.Tutor.Id))
    .ForMember(d => d.TutorName, o => o.MapFrom(x => x.Assignment.Tutor.User.FullName()))
    .ForMember(d => d.AssignmentTitle, o => o.MapFrom(x => x.Assignment.Title))
    .ForMember(d => d.Description, o => o.MapFrom(x => x.Assignment.Description))
    .ForMember(d => d.SubjectName, o => o.MapFrom(x => x.Assignment.Subject.Name))
    .ForMember(d => d.AssignmentFiles, opt => opt.MapFrom(s => s.Assignment.Files));
```

```
        //Review and Feedbacks
        CreateMap<Review, ReviewModel>();
        CreateMap<ReviewModel, Review>()
            .ForMember(d => d.TutorName, o => o.MapFrom(x => x.Tutor.User.FullName()))
            .ForMember(d => d.AuthorName, o => o.MapFrom(x => x.Author.FullName()));
    }
}
```

Файл TimeType.cs

```
namespace Domain.Helpers;

public enum TimeType
{
    Available, //  тільки з налаштувань Tutor
    Busy, //  мають іншу зайнятість
    Unavailable //  У цей час репетитор не працює
}
```

Файл Assignment.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Http;

namespace Domain.Models;

public class Assignment
{
    [DisplayName("#")] public int Id { get; set; }
    [Range(1, int.MaxValue)] public int TutorId { get; set; }

    [DisplayName("Вчитель")] public string TutorName { get; set; } = string.Empty;

    [DisplayName("Предмет")]
    [Range(1, int.MaxValue, ErrorMessage = "Оберіть предмет")]
    public int SubjectId { get; set; }

    [DisplayName("Предмет")] public string? SubjectName { get; set; } = null;

    [DisplayName("Назва завдання")]
    [StringLength(50, ErrorMessage = "{0} має містити принаймні {2} і не більше {1} символів.", MinimumLength =
3)]
    public string Title { get; set; } = string.Empty;

    [DisplayName("Опис завдання")]
    [StringLength(500, ErrorMessage = "{0} має містити принаймні {2} і не більше {1} символів.", MinimumLength =
0)]
    public string? Description { get; set; } = string.Empty;

    [DisplayName("Термін здачі")] public DateTime Deadline { get; set; } = DateTime.Today.AddDays(7).Date;

    [DisplayName("Файли до завдання")]
    public List<UserFile> FileNames { get; set; } = [];


    [DisplayName("Учні")] public List<int> StudentsIds { get; set; } = [];

    [DisplayName("Учні")] public string StudentNames { get; set; } = string.Empty;
    [DisplayName("Додано")] public DateTime CreatedAt { get; set; }
    [DisplayName("Оновлено")] public DateTime? UpdatedAt { get; set; }
}
```

Файл File.cs

```
using System.ComponentModel.DataAnnotations;
using Infra.DatabaseAdapter.Models;
```

```
using Microsoft.EntityFrameworkCore.Metadata.Internal;
using Microsoft.VisualBasic.CompilerServices;

namespace Domain.Models;

public class UserFile
{
   [MaxLength(36)] public Guid Id { get; set; }
   [MaxLength(100)] public string FileName { get; set; } = string.Empty;

   public int OwnerId { get; set; }
   public string OwnerName { get; set; } = null!;
   public bool IsOwner { get; set; }
}
```

Файл Lesson.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Domain.Models;

public class Lesson
{
   public int Id { get; set; }
   [Range(1, int.MaxValue)] public int TutorId { get; set; }


   [DisplayName("Вчитель")] public string TutorName { get; set; } = string.Empty;

   [DisplayName("Початок")]
   [DisplayFormat(DataFormatString = "{0:HH:mm}")]
   public DateTime From { get; set; }

   [DisplayName("Кінець")]
   [DisplayFormat(DataFormatString = "{0:HH:mm}")]
   public DateTime To { get; set; }

   [StringLength(254, ErrorMessage = "{0} має містити не більше {1} символів.")]
   [DisplayName("Коментар")]
   public string? Comment { get; set; } = string.Empty;

   [DisplayName("Предмет")]
   [Range(1, int.MaxValue, ErrorMessage = "Оберіть предмет")]
   public int SubjectId { get; set; }

   [DisplayName("Предмет")] public string? SubjectName { get; set; } = null;

   [DisplayName("Учні")] public List<int> StudentsIds { get; set; } = [];

   [DisplayName("Учні")] public string StudentNames { get; set; } = string.Empty;

   [DisplayName("Дата")]
   [DisplayFormat(DataFormatString = "{0:d}")]
   public DateOnly LessonDate => DateOnly.FromDateTime(From);
}
```

Файл LessonRequest.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Globalization;
using AutoMapper.Configuration.Annotations;
using Infra.DatabaseAdapter.Helpers;

namespace Domain.Models;
```

```
public class LessonRequest
{
    [DisplayName("#")] public int Id { get; set; }

    [DisplayName("Предмет")] public string Subject { get; set; } = null!;
    [DisplayName("Учень")] public string UserName { get; set; } = string.Empty;
    [DisplayName("Вчитель")] public string TutorName { get; set; } = string.Empty;

    [StringLength(254, ErrorMessage = "{0} має містити не більше {1} символів.", MinimumLength = 0)]
    [DisplayName("Посилання на заняття")] public string TutorComment { get; set; } = string.Empty;
    [DisplayName("Коментар учня")] public string Comment { get; set; } = string.Empty;


    [DisplayName("Статус")] public LessonRequestStatus Status { get; set; }

    [DisplayName("Початок")]
    [DisplayFormat(DataFormatString = "{0:HH:mm}")]
    public DateTime From { get; set; }


    [DisplayName("Кінець")]
    [DisplayFormat(DataFormatString = "{0:HH:mm}")]

    public DateTime To { get; set; }

    public bool IsTutor { get; set; }
    public int TutorId { get; set; }
    public int CreatedId { get; set; }


    [DisplayName("Дата")]
    [DisplayFormat(DataFormatString = "{0:d}")]
    public DateOnly LessonDate => DateOnly.FromDateTime(From);
}
```

Файл LessonSession.cs

```
using Domain.Helpers;

namespace Domain.Models;

public class LessonSession
{
    public int Id { get; set; }
    public TimeType Type { get; set; }
    public string Title { get; set; } = "";
    public DateTime From { get; set; }
    public DateTime To { get; set; }
}
```

Файл Review.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Domain.Models;

public class Review
{
    public int Id { get; set; }
    public int TutorId { get; set; }
    [DisplayName("Вчитель")] public string TutorName { get; set; } = null!;

    [Range(1, 10, ErrorMessage = "Оцінка обов'язкова")]
    [DisplayName("Оцінка")]
    [Required(ErrorMessage = "Оцінка обов'язкова")]
```

```
    public int Rating { get; set; }

    [DisplayName("Опис")]
    [Required(ErrorMessage = "Введіть коментар")]
    [StringLength(1000, ErrorMessage = "{0} має містити не більше {1} символів.", MinimumLength = 0)]
    public string Description { get; set; } = string.Empty;

    public int AuthorId { get; set; }
    [DisplayName("Автор")] public string AuthorName { get; set; } = null!;

    //ITrackable
    [DisplayName("Створено")] public DateTime CreatedAt { get; set; }
    [DisplayName("Оновлено")] public DateTime? UpdatedAt { get; set; }

    public string Avatar => "/avatars/" + AuthorId + ".png";
}
```

Файл Solution.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;

namespace Domain.Models;

public class Solution
{
    //Solution Model
    public int Id { get; set; }
    public int AssignmentId { get; set; }
    public int TutorId { get; set; }

    public int StudentId { get; set; }
    [DisplayName("Учень")] public string StudentName { get; set; } = string.Empty;
    [DisplayName("Статус")] public SolutionStatus Status { get; set; }

    [DisplayName("Відповідь")]
    [MaxLength(500)]
    public string? Answer { get; set; }

    [DisplayName("Коментар вчителя")]
    [MaxLength(500)]
    public string? TutorComment { get; set; }

    [DisplayName("Файли до завдання")] public List<UserFile> SolutionFiles { get; set; } = [];

    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }

    //Assignment Model
    [MaxLength(50)]
    [DisplayName("Вчитель")]

    public string TutorName { get; set; } = string.Empty;

    [DisplayName("Предмет")] public string SubjectName { get; set; } = string.Empty;

    [DisplayFormat(DataFormatString = "{0,20}")]
    [MaxLength(50)]
    [DisplayName("Назва")]
    public string AssignmentTitle { get; set; } = string.Empty;

    [DisplayName("Опис завдання")]
    [StringLength(500, ErrorMessage = "{0} має містити принаймні {2} і не більше {1} символів.", MinimumLength =
0)]
    public string? Description { get; set; } = string.Empty;
```

```
    [DisplayName("Термін здачі")]
    public DateOnly Deadline { get; set; } = DateOnly.FromDateTime(DateTime.Today.AddDays(7));

    [DisplayName("Файли відповіді")] public List<UserFile> AssignmentFiles { get; set; } = [];
}
```

Файл Tutor.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Domain.Models;

public class Tutor
{
    public int Id { get; set; }


    [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
    [Display(Name = "Про себе")]
    [StringLength(3000, ErrorMessage = "Введені дані не можуть бути довшими за 3000 символів")]
    [MinLength(50, ErrorMessage = "Занадто короткий текст")]
    public string About { get; set; } = string.Empty;

    [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
    [Display(Name = "Домашня адреса")]
    [StringLength(300, ErrorMessage = "Введені дані не можуть бути довшими за 300 символів.")]
    public string Address { get; set; } = string.Empty;

    [DisplayName("У вчителя")] public bool TutorHomeAccess { get; set; }
    [DisplayName("В учня")] public bool StudentHomeAccess { get; set; }
    [DisplayName("Онлайн")] public bool OnlineAccess { get; set; }

    [DisplayName("Коротний опис")]
    [StringLength(254, ErrorMessage = "Введені дані не можуть бути довшими за 254 символів.")]
    [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
    public string Descriptions { get; set; } = string.Empty;

    [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
    [DisplayName("Ціна за одну годину")]
    [DataType(DataType.Currency)]
    [Range(10, 9999)]
    public int HourRate { get; set; }

    [Display(Name = "Оберіть предмети які плануєте викладати")]
    public List<int> SubjectIds { get; set; } = [];

    public int ReviewCount { get; set; }
    public int ReviewValue { get; set; }

    public int LessonCount { get; set; }
}
```

Файл User.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Runtime.InteropServices;
using System.Runtime.Serialization;
using Microsoft.AspNetCore.Http;

namespace Domain.Models;

public class User
{
    public int Id { get; set; }
```

```
    [DisplayName("Активувати профіль викладача")]
    public bool ProfileEnabled { get; set; }

    [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
    [Display(Name = "Ім'я")]
    [StringLength(50, ErrorMessage = "Надто довгий текст у полі ім'я", MinimumLength = 3)]
    public string Name { get; set; } = string.Empty;

    public string Avatar => "/avatars/" + Id + ".png";

    [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
    [Display(Name = "Прізвище")]
    [StringLength(50, ErrorMessage = "Надто довгий текст у полі прізвище", MinimumLength = 3)]
    public string Surname { get; set; } = string.Empty;

    [MaxLength(50, ErrorMessage = "Надто довгий текст у полі по батькові")]
    [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
    [Display(Name = "По батькові")]
    public string Patronymic { get; set; } = string.Empty;

    [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
    [Display(Name = "Місто")]
    public string CityId { get; set; } = "0";

    public string FullName { get; set; } = string.Empty;
    public string CityName { get; set; } = string.Empty;
}
```

Файл CheckFavoriteQuery.cs

```
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class CheckFavoriteQuery : IRequest<bool>
{
    public int TutorId { get; set; }
    public int UserId { get; set; }

    public class CheckFavoriteQueryHandler : BaseMediatrHandler<CheckFavoriteQuery, bool>
    {
        public override async Task<bool> Handle(CheckFavoriteQuery r, CancellationToken token)
        {
            var tutorSaved = await DatabaseContext.Users.AsNoTracking().AnyAsync(x =>
                x.Id == r.UserId &&
                x.FavoriteTutors.Any(t => t.Id == r.TutorId));
            return tutorSaved;
        }

        public CheckFavoriteQueryHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл DownloadFileQuery.cs

```
using AutoMapper;
```

```
using Domain.Helpers;
using Infra;
using Infra.DatabaseAdapter;
using Infra.Storage;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace Domain.Queries;

public class DownloadFileQuery : IRequest<StorageFile>
{
   public string FileId { get; set; } = string.Empty;
   public int UserId { get; set; }

   public class DownloadFileQueryHandler : BaseMediatrHandler<DownloadFileQuery, StorageFile>
   {
      private readonly IStorageRepository _storageRepo;


      public DownloadFileQueryHandler(AppDbContext dbContext, IMapper mapper, IStorageRepository storageRepo) :
         base(dbContext, mapper) => _storageRepo = storageRepo;


      public override async Task<StorageFile> Handle(DownloadFileQuery r, CancellationToken token)
      {
         var dbFile = await DatabaseContext.Files.AsNoTracking().FirstOrDefaultAsync(x => x.Id.ToString() == r.FileId);
         if (dbFile == null)
            throw new FileNotFoundException("Файл не знайдено у базі даних");

         var isOwner = dbFile.OwnerId == r.UserId && r.UserId != 0;
         var studentAccess = await DatabaseContext.Files.AsNoTracking() //доступ до студентів до файлів вчителів
            .Include(x => x.Assignments)
            .ThenInclude(x => x.Solutions)
            .AnyAsync(f => f.Assignments.Any(a => a.Solutions.Any(s => s.StudentId == r.UserId)));
         var tutorAccess = await DatabaseContext.Files.AsNoTracking() // доступ вчителів для файлів студентів
            .Include(x => x.Solutions)
            .ThenInclude(x => x.Assignment)
            .AnyAsync(f => f.Solutions.Any(s => s.Assignment.TutorId == r.UserId));

         if (!(isOwner || studentAccess || tutorAccess)) // якщо користувач не має ні одного типу доступу
            throw new AccessDeniedException("Доступ до файлу обмежений");

         var file = _storageRepo.GetFile(dbFile.ServerName);
         return file;
      }
   }
}
```

Файл GetAllCitiesQuery.cs

```
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetAllCitiesQuery : IRequest<Dictionary<int, string>>
{
   public class GetAllCitiesQueryHandler : BaseMediatrHandler<GetAllCitiesQuery, Dictionary<int, string>>
   {
      public GetAllCitiesQueryHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
```

```
        {
        }


        public override async Task<Dictionary<int, string>> Handle(GetAllCitiesQuery r, CancellationToken token) =>
            await DatabaseContext.Cities.AsNoTracking().ToDictionaryAsync(k => k.Id, v => v.FullName());
    }
}
```

Файл GetAllSubjectsQuery.cs

```
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetAllSubjectsQuery : IRequest<Dictionary<int, string>>
{
    public int? TutorId { get; set; }

    public class GetAllSubjectsQueryHandler : BaseMediatrHandler<GetAllSubjectsQuery, Dictionary<int, string>>
    {
        public GetAllSubjectsQueryHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }

        public override async Task<Dictionary<int, string>> Handle(GetAllSubjectsQuery r, CancellationToken token)
        {
            List<SubjectModel> subjects;
            if (r.TutorId.HasValue)
                subjects = await DatabaseContext.Tutors
                    .Include(x => x.Subjects)
                    .AsNoTracking()
                    .Where(x => x.Id == r.TutorId.Value)
                    .SelectMany(x => x.Subjects)
                    .ToListAsync();
            else
                subjects = await DatabaseContext.Subjects.AsNoTracking().ToListAsync();

            return subjects.ToDictionary(k => k.Id, v => v.Name);
        }
    }
}
```

Файл GetAssignmentNamesQuery.cs

```
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetAssignmentNamesQuery : IRequest<Dictionary<int, string>>
{
    public int? TutorId { get; set; }
    public int? StudentId { get; set; }

    public class GetAssignmentNamesQueryHandler : BaseMediatrHandler<GetAssignmentNamesQuery, Dictionary<int,
string>>
```

```
    {
        public override async Task<Dictionary<int, string>> Handle(GetAssignmentNamesQuery r, CancellationToken
token)
        {
            if (!r.TutorId.HasValue && !r.StudentId.HasValue)
                throw new CommandParameterException("Не вказані параметри запиту");

            var q = DatabaseContext.Assignments.AsNoTracking()
                .Include(x => x.Solutions)
                .Where(x =>
                    (r.TutorId.HasValue && r.TutorId == x.TutorId) || //Пошук завдань як вчителя
                    (r.StudentId.HasValue && x.Solutions.Any(s => s.StudentId == r.StudentId)) //Пошук як учня
                ).AsQueryable();
            return await q.ToDictionaryAsync(k => k.Id, v => v.Title);
        }

        public GetAssignmentNamesQueryHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл GetAssignmentsQuery.cs

```
using System.ComponentModel;
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Helpers;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Queries;

public class GetAssignmentsQuery : IRequest<List<Assignment>>
{
    public int UserId { get; set; }
    [DisplayName("Предмет")] public List<int> SubjectId { get; set; } = [];
    [DisplayName("Учень")] public List<int> StudentId { get; set; } = [];


    public class GetAssignmentsQueryHandler : BaseMediatrHandler<GetAssignmentsQuery, List<Assignment>>
    {
        public override async Task<List<Assignment>> Handle(GetAssignmentsQuery r, CancellationToken token)
        {
            //Запит
            var q = DatabaseContext.Assignments.AsNoTracking()
                .Include(x => x.Subject)
                .Include(x => x.Solutions)
                .Include(x => x.Files)
                .ThenInclude(x => x.Owner)
                .Include(x => x.Tutor)
                .ThenInclude(x => x.User)
                .Where(x => x.TutorId == r.UserId)
                .AsQueryable();

            //Filters
            if (r.SubjectId.Count > 0)
                q = q.Where(x => r.SubjectId.Contains(x.SubjectId));
            if (r.StudentId.Count > 0)
                q = q.Where(x => x.Solutions.Any(s => r.StudentId.Contains(s.StudentId)));

            //Execute
            var dbAssignments = await q.ToListAsync();
```

```
            var assignments = Mapper.Map<List<Assignment>>(dbAssignments);
            return assignments;
        }

        public GetAssignmentsQueryHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл GetEventQuery.cs

```
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Queries;

public class GetEventQuery : IRequest<List<LessonSession>>
{
    public List<TimeType> TimeTypes { get; set; } = [];

    public int UserId { get; set; }
    public DateTime? From { get; set; }
    public DateTime? To { get; set; }

    public class GetEventQueryHandler : BaseMediatrHandler<GetEventQuery, List<LessonSession>>
    {
        public override async Task<List<LessonSession>> Handle(GetEventQuery r, CancellationToken token)
        {
            if (!r.From.HasValue) r.From = DateTime.MinValue;
            if (!r.To.HasValue) r.To = DateTime.MaxValue;

            List<LessonSession> events = new();
            var dbLessons = await DatabaseContext.Lessons.AsNoTracking()
                .Where(x => x.Students.Any(y => y.Id == r.UserId) || x.TutorId == r.UserId)
                .Where(x => r.From <= x.To && x.From <= r.To)
                .ToListAsync();

            var dbAvailableTime = await DatabaseContext.AvailableTimes.AsNoTracking()
                .Where(x => x.ProfileId == r.UserId)
                .OrderBy(x => x.DayOfWeek)
                .ThenBy(x => x.StartTime)
                .ToListAsync();

            //Calc Busy Time
            foreach (var lesson in dbLessons)
                events.Add(new LessonSession
                {
                    Id = lesson.Id,
                    Type = TimeType.Busy,
                    Title = lesson.Title,
                    From = lesson.From > r.From.Value ? lesson.From : r.From.Value,
                    To = lesson.To < r.To.Value ? lesson.To : r.To.Value
                });

            //Available Time
            var prevSunday = DateTime.Today.AddDays(-(int)DateTime.Today.DayOfWeek);

            var availableTimes = new List<LessonSession>();
            for (var i = -1; i <= 4; i++) //Розрахунок наступних тижнів 4
            {
```

```
            var offsetWeek = i * 7;
            foreach (var dbRange in dbAvailableTime)
                availableTimes.Add(new LessonSession
                {
                    Id = dbRange.Id,
                    Type = TimeType.Available,
                    From = prevSunday.AddDays(offsetWeek + dbRange.DayOfWeek).AddTicks(dbRange.StartTime.Ticks),
                    To = prevSunday.AddDays(offsetWeek + dbRange.DayOfWeek).AddTicks(dbRange.EndTime.Ticks)
                });
        }

        availableTimes = availableTimes.Where(x => r.From <= x.From && x.To <= r.To).ToList();
        events.AddRange(availableTimes);

        events = events.OrderBy(x => x.From).ToList();

        //Unavailable Time
        var unavailableTimes = new List<LessonSession>();
        var prevEventEnd = r.From.Value; // Фиксування почутку дня

        //коригування на початок тижня
        if (availableTimes.Count > 0 && availableTimes[0].From != r.From)
            unavailableTimes.Add(new LessonSession()
            {
                Type = TimeType.Unavailable,
                From = r.From.Value,
                To = availableTimes[0].From
            });


        for (var i = 0; i < availableTimes.Count; i++)
        {
            if (i != 0 && availableTimes[i - 1].From.Day != availableTimes[i].From.Day) //Якщо дата не та
                prevEventEnd = availableTimes[i - 1].To; // Взяти останній час з минулого дня

            if ((availableTimes[i].From - prevEventEnd).TotalMinutes != 0) //Якщо події не йдуть поспіль
                unavailableTimes.Add(new LessonSession
                {
                    Type = TimeType.Unavailable,
                    From = prevEventEnd,
                    To = availableTimes[i].From
                });

            prevEventEnd = availableTimes[i].To;
        }


        //коригування на кінець тижня
        if (availableTimes.Count == 0 || availableTimes[availableTimes.Count - 1].To != r.To)
            unavailableTimes.Add(new LessonSession()
            {
                Type = TimeType.Unavailable,
                From = prevEventEnd,
                To = r.To.Value
            });

        events.AddRange(unavailableTimes);

        //Type Filter
        if (r.TimeTypes.Count > 0)
            events = events.Where(x => r.TimeTypes.Contains(x.Type)).ToList();

        return events.ToList();
    }

    public GetEventQueryHandler(AppDbContext dbContext, IMapper mapper)
        : base(dbContext, mapper)
```

```
        {
        }
    }
}
```

Файл GetLessonsQuery.cs

```
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Queries;

public class GetLessonsQuery : IRequest<List<Lesson>>
{
    public int? StudentId { get; set; }
    public int? TutorId { get; set; }
    public DateTime? From { get; set; }
    public DateTime? To { get; set; }

    public class GetLessonsQueryHandler : BaseMediatrHandler<GetLessonsQuery, List<Lesson>>
    {
        public override async Task<List<Lesson>> Handle(GetLessonsQuery r, CancellationToken token)
        {
            //Підготовка
            if (!r.TutorId.HasValue && !r.StudentId.HasValue)
                throw new CommandParameterException("Не вказані параметри запиту");

            if (!r.From.HasValue)
                r.From = DateTime.MinValue;
            if (!r.To.HasValue)
                r.To = DateTime.MaxValue;

            //Запит
            var dbLessons = await DatabaseContext.Lessons.AsNoTracking()
                .Include(x => x.Subject)
                .Include(x => x.Students)
                .Include(x => x.Tutor)
                .ThenInclude(x => x.User)
                .Where(x => x.To > r.From || r.To > x.From)
                .Where(x =>
                    (r.StudentId.HasValue && x.Students.Any(s => s.Id == r.StudentId)) || //Взтяти всі записи як учня
                    (r.TutorId.HasValue && x.TutorId == r.TutorId)) //Додати всі записи як вчителя
                .ToListAsync();

            var lesList = Mapper.Map<List<Lesson>>(dbLessons);
            return lesList;
        }

        public GetLessonsQueryHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл GetOneAssignmentQuery.cs

```
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
```

```
using Domain.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetOneAssignmentQuery : IRequest<Assignment?>
{
   public int AssignmentId { get; set; }
   public int UserId { get; set; }

   public class GetOneAssignmentQueryHandler : BaseMediatrHandler<GetOneAssignmentQuery, Assignment?>
   {
      public override async Task<Assignment?> Handle(GetOneAssignmentQuery r, CancellationToken token)
      {
         //Запит
         var dbTask = await DatabaseContext.Assignments.AsNoTracking()
            .Include(x => x.Subject)
            .Include(x => x.Solutions)
            .Include(x => x.Files)
            .ThenInclude(x => x.Owner)
            .Include(x => x.Tutor)
            .ThenInclude(x => x.User)
            .Where(x => x.TutorId == r.UserId || x.Solutions.Any(y => y.StudentId == r.UserId))
            .FirstOrDefaultAsync(x => x.Id == r.AssignmentId);

         if (dbTask == null)
            throw new AssignmentException("Задачу не знайдено");

         var assignment = Mapper.Map<Assignment>(dbTask);
         return assignment;
      }

      public GetOneAssignmentQueryHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
      {
      }
   }
}
```

Файл GetOneLessonQuery.cs

```
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetOneLessonQuery : IRequest<Lesson?>
{
   public int LessonId { get; set; }
   public int UserId { get; set; }

   public class GetOneLessonQueryHandler : BaseMediatrHandler<GetOneLessonQuery, Lesson?>
   {
      public override async Task<Lesson?> Handle(GetOneLessonQuery r, CancellationToken token)
      {
         //Запит
         var dbLesson = await DatabaseContext.Lessons.AsNoTracking()
            .Include(x => x.Students)
            .Include(x => x.Subject)
            .Include(x => x.Tutor)
```

```
            .ThenInclude(x => x.User)
            .FirstOrDefaultAsync(x =>
                (x.TutorId == r.UserId || x.Students.Any(s => s.Id == r.UserId))
                && x.Id == r.LessonId);
        if (dbLesson == null)
            return null;

        var lesson = Mapper.Map<Lesson>(dbLesson);
        return lesson;
    }

    public GetOneLessonQueryHandler(AppDbContext dbContext, IMapper mapper)
        : base(dbContext, mapper)
    {
    }
  }
}
```

Файл GetOneSolutionQuery.cs

```
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetOneSolutionQuery : IRequest<Solution?>
{
  public int SolutionId { get; set; }
  public int UserId { get; set; }

  public class GetOneSolutionQueryHandler : BaseMediatrHandler<GetOneSolutionQuery, Solution?>
  {
    public override async Task<Solution?> Handle(GetOneSolutionQuery r, CancellationToken token)
    {
      //Запит
      var dbTask = await DatabaseContext.Solutions.AsNoTracking()
        .Include(x => x.Student)
        .Include(x => x.Assignment)
        .ThenInclude(x => x.Tutor)
        .ThenInclude(x => x.User)
        .Include(x => x.Assignment.Subject)
        .Include(x => x.Files)
        .ThenInclude(x => x.Owner)
        .Include(x => x.Assignment.Files)
        .Where(x => x.StudentId == r.UserId || x.Assignment.TutorId == r.UserId)
        .FirstOrDefaultAsync(x => x.Id == r.SolutionId);

      if (dbTask == null)
        throw new SolutionException("Задачу не знайдено");

      var solution = Mapper.Map<Solution>(dbTask);
      return solution;
    }

    public GetOneSolutionQueryHandler(AppDbContext dbContext, IMapper mapper)
      : base(dbContext, mapper)
    {
    }
  }
}
```

Файл GetReviewQuery.cs

```
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetReviewQuery : IRequest<Review>
{
   public int AuthorId { get; set; }
   public int TutorId { get; set; }

   public class GetReviewQueryHandler : BaseMediatrHandler<GetReviewQuery, Review>
   {
      public override async Task<Review> Handle(GetReviewQuery r, CancellationToken token)
      {
         if (r.AuthorId == 0 || r.TutorId == 0)
            throw new CommandParameterException("Ідентіфикатори невірні");

         //Запит
         var dbReview = await DatabaseContext.Reviews.AsNoTracking()
            .Include(x => x.Author)
            .Include(x => x.Tutor)
            .ThenInclude(x => x.User)
            .FirstOrDefaultAsync(x => x.TutorId == r.TutorId && r.AuthorId == x.AuthorId);

         // Повернути новій якщо відгук не знайдено
         Review review;
         if (dbReview == null)
         {
            review = new Review() { AuthorId = r.AuthorId, TutorId = r.TutorId };
            review.Rating = 10;
            review.TutorName = await DatabaseContext.Users.AsNoTracking()
                        .Where(x => x.Id == r.TutorId)
                        .Select(x => x.FullName()).FirstOrDefaultAsync() ??
                     throw new ReviewException("Ідентифікатор вчителя не знайдено", r.TutorId);
            review.AuthorName = await DatabaseContext.Users.AsNoTracking()
                         .Where(x => x.Id == r.AuthorId)
                         .Select(x => x.FullName()).FirstOrDefaultAsync() ??
                      throw new ReviewException("Ідентифікатор автора не знайдено", r.AuthorId);
         }
         else
         {
            review = Mapper.Map<Review>(dbReview);
         }

         return review;
      }

      public GetReviewQueryHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
      {
      }
   }
}
```

Файл GetSolutionsQuery.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using AutoMapper;
```

```
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Queries;

public class GetSolutionsQuery : IRequest<List<Solution>>
{
   public int UserId { get; set; }

   [DisplayName("Статус задачі")] public List<SolutionStatus> Status { get; set; } = [];

   [DisplayName("Предмет")] public List<int> SubjectId { get; set; } = [];
   [DisplayName("Учень")] public List<int> StudentId { get; set; } = [];
   [DisplayName("Вчитель")] public List<int> TutorId { get; set; } = [];
   [DisplayName("Задача")] public List<int> AssignmentId { get; set; } = [];

   public class GetSolutionsQueryHandler : BaseMediatrHandler<GetSolutionsQuery, List<Solution>>
   {
      public override async Task<List<Solution>> Handle(GetSolutionsQuery r, CancellationToken token)
      {
         //Запит
         var q = DatabaseContext.Solutions.AsNoTracking()
            .Include(x => x.Student)
            .Include(x => x.Assignment)
            .ThenInclude(x => x.Tutor)
            .ThenInclude(x => x.User)
            .Include(x => x.Assignment.Subject)
            .Include(x => x.Files)
            .ThenInclude(x => x.Owner)
            .Where(x => x.StudentId == r.UserId || x.Assignment.TutorId == r.UserId)
            .AsQueryable();
         //Filters
         if (r.SubjectId.Count > 0)
            q = q.Where(x => r.SubjectId.Contains(x.Assignment.SubjectId));
         if (r.Status.Count > 0)
            q = q.Where(x => r.Status.Contains(x.Status));
         if (r.StudentId.Count > 0)
            q = q.Where(x => r.StudentId.Contains(x.StudentId));
         if (r.TutorId.Count > 0)
            q = q.Where(x => r.TutorId.Contains(x.Assignment.TutorId));
         if (r.AssignmentId.Count > 0)
            q = q.Where(x => r.AssignmentId.Contains(x.AssignmentId));

         //Execute
         var dbSolutions = await q.ToListAsync();
         var solutions = Mapper.Map<List<Solution>>(dbSolutions);
         return solutions;
      }

      public GetSolutionsQueryHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
      {
      }
   }
}
```

Файл GetStudentTutorsQuery.cs

```
using AutoMapper;
using Domain.Helpers;
```

```
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetStudentTutorsQuery : IRequest<Dictionary<int, string>>
{
   public int StudentId { get; set; }

   public class GetStudentTutorsQueryHandler : BaseMediatrHandler<GetStudentTutorsQuery, Dictionary<int, string>>
   {
      public GetStudentTutorsQueryHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
      {
      }

      public override async Task<Dictionary<int, string>> Handle(GetStudentTutorsQuery r, CancellationToken token)
      {
         //Вибір вчителів у яких були уроки зі студентом
         var students = await DatabaseContext.Lessons.AsNoTracking()
            .Include(x => x.Tutor)
            .ThenInclude(x => x.User)
            .Where(x => x.Students.Any(s => s.Id == r.StudentId))
            .GroupBy(x => x.TutorId)
            .Select(x => x.Key)
            .Join(DatabaseContext.Users, x => x, u => u.Id, (_, u) => u)
            .ToDictionaryAsync(g => g.Id, g => g.FullName());
         return students;
      }
   }
}
```

Файл GetTutorProfileQuery.cs

```
using System.Diagnostics;
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Queries;

public class GetTutorProfileQuery : IRequest<Tutor>
{
   public int ProfileId { get; set; }

   public class GetTutorProfileQueryHandler : BaseMediatrHandler<GetTutorProfileQuery, Tutor>
   {
      public override async Task<Tutor> Handle(GetTutorProfileQuery r, CancellationToken token)
      {
         var dbProfile = await DatabaseContext.Tutors.AsNoTracking()
            .Include(x => x.About)
            .Include(x => x.Subjects)
            .Include(x => x.Reviews)
            .Include(x => x.TeachingLessons)
            .FirstOrDefaultAsync(x => x.Id == r.ProfileId);

         //Перевірка на існування  профілю
         if (dbProfile == null)
            if (await DatabaseContext.Users.AsNoTracking().AnyAsync(x => x.Id == r.ProfileId))
               return new Tutor(); //Новий профіль
            else
```

```
                throw new UserNotFoundException("Профіль вчителя не знайдено");

            var profile = Mapper.Map<Tutor>(dbProfile);
            if (dbProfile.Reviews.Count > 0)
                profile.ReviewValue = (int)dbProfile.Reviews.Average(x => x.Rating);
            return profile;
        }

        public GetTutorProfileQueryHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл GetTutorProfileQuery.cs

```
using System.Diagnostics;
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Queries;

public class GetTutorProfileQuery : IRequest<Tutor>
{
    public int ProfileId { get; set; }

    public class GetTutorProfileQueryHandler : BaseMediatrHandler<GetTutorProfileQuery, Tutor>
    {
        public override async Task<Tutor> Handle(GetTutorProfileQuery r, CancellationToken token)
        {
            var dbProfile = await DatabaseContext.Tutors.AsNoTracking()
                .Include(x => x.About)
                .Include(x => x.Subjects)
                .Include(x => x.Reviews)
                .Include(x => x.TeachingLessons)
                .FirstOrDefaultAsync(x => x.Id == r.ProfileId);

            //Перевірка на існування  профілю
            if (dbProfile == null)
                if (await DatabaseContext.Users.AsNoTracking().AnyAsync(x => x.Id == r.ProfileId))
                    return new Tutor(); //Новий профіль
                else
                    throw new UserNotFoundException("Профіль вчителя не знайдено");

            var profile = Mapper.Map<Tutor>(dbProfile);
            if (dbProfile.Reviews.Count > 0)
                profile.ReviewValue = (int)dbProfile.Reviews.Average(x => x.Rating);
            return profile;
        }

        public GetTutorProfileQueryHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл GetTutorReviewsQuery.cs

```
using Domain.Models;
```

```
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetTutorReviewsQuery : IRequest<List<Review>>
{
   public int TutorId { get; set; }

   public class GetTutorReviewsQueryHandler : BaseMediatrHandler<GetTutorReviewsQuery, List<Review>>
   {
      public override async Task<List<Review>> Handle(GetTutorReviewsQuery r, CancellationToken token)
      {
         if (r.TutorId == 0)
            throw new UserNotFoundException("Ідентіфикатори невірні");

         //Запит
         var dbReview = await DatabaseContext.Reviews.AsNoTracking()
            .Include(x => x.Tutor)
            .Include(x => x.Author)
            .Where(x => x.TutorId == r.TutorId).ToListAsync();

         var reviews = Mapper.Map<List<Review>>(dbReview);
         return reviews;
      }

      public GetTutorReviewsQueryHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
      {
      }
   }
}
```

Файл GetTutorsQuery.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Linq.Expressions;
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetTutorsQuery : IRequest<List<int>>
{
   public int? HourRateFrom { get; set; }
   public int? HourRateTo { get; set; }
   public string CityId { get; set; } = "0";

   public string SubjectId { get; set; } = "0";
   public bool OnlineAccess { get; set; }
   public bool TutorHomeAccess { get; set; }
   public bool StudentHomeAccess { get; set; }

   public string? SearchText { get; set; } = string.Empty;
```

```
    public int IdentityId { get; set; }
    public bool OnlyFavorites { get; set; }

    public class GetTutorsQueryHandler : BaseMediatrHandler<GetTutorsQuery, List<int>>
    {
        public override async Task<List<int>> Handle(GetTutorsQuery r, CancellationToken token)
        {
            var q = DatabaseContext.Users.AsNoTracking()
                .Include(x => x.Tutor)
                .ThenInclude(x => x.Subjects)
                .Include(x => x.Tutor.InFavorite)
                .Where(x => x.ProfileEnabled);

            if (r.HourRateFrom != null)
                q = q.Where(x => x.Tutor.HourRate >= r.HourRateFrom);
            if (r.HourRateTo != null)
                q = q.Where(x => x.Tutor.HourRate <= r.HourRateTo);
            if (r.CityId != "0")
                q = q.Where(x => x.CityId == int.Parse(r.CityId));
            if (r.SubjectId != "0")
                q = q.Where(x => x.Tutor.Subjects.Any(s => s.Id == int.Parse(r.SubjectId)));
            if (r.OnlineAccess)
                q = q.Where(x => x.Tutor.OnlineAccess);
            if (r.TutorHomeAccess)
                q = q.Where(x => x.Tutor.TutorHomeAccess);
            if (r.StudentHomeAccess)
                q = q.Where(x => x.Tutor.StudentHomeAccess);
            if (r.OnlyFavorites)
                q = q.Where(x => x.Tutor.InFavorite.Any(f => f.Id == r.IdentityId));

            List<int> tutorsIds;
            if (r.SearchText != null && r.SearchText != string.Empty)
            {
                var searchText = r.SearchText.ToUpperInvariant().Trim().Split(' ');
                foreach (var partSearch in searchText)
                    q = q.Where(x => x.NormalizeName.Contains(partSearch));
            }

            tutorsIds = await q.OrderBy(x => x.Reviews.Count).Select(x => x.Id).ToListAsync();
            return tutorsIds;
        }

        public GetTutorsQueryHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл GetTutorStudentsQuery.cs

```
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Queries;

public class GetTutorStudentsQuery : IRequest<Dictionary<int, string>>
{
    public int TutorId { get; set; }

    public class GetTutorStudentsQueryHandler : BaseMediatrHandler<GetTutorStudentsQuery, Dictionary<int, string>>
    {
        public GetTutorStudentsQueryHandler(AppDbContext dbContext, IMapper mapper)
```

```
            : base(dbContext, mapper)
        {
        }

        public override async Task<Dictionary<int, string>> Handle(GetTutorStudentsQuery r, CancellationToken token)
        {
            //Вибір усіх студентів, у яких був хоча б один урок з репетитором
            var students = await DatabaseContext.Requests.AsNoTracking()
                .Include(x => x.Created)
                .Where(x => x.TutorId == r.TutorId)
                .GroupBy(x => x.Created)
                .Select(x => x.Key)
                .ToDictionaryAsync(x => x.Id, x => x.FullName());
            return students;
        }
    }
}
```

Файл GetUserProfileQuery.cs

```
using AutoMapper;
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace Domain.Queries;

public class GetUserProfileQuery : IRequest<User>
{
    public int ProfileId { get; set; }

    public class GetUserProfileQueryHandler : BaseMediatrHandler<GetUserProfileQuery, User>
    {
        public override async Task<User> Handle(GetUserProfileQuery r, CancellationToken token)
        {
            var dbProfile = await DatabaseContext.Users.AsNoTracking()
                .Include(x => x.City)
                .FirstOrDefaultAsync(x => x.Id == r.ProfileId);

            //Перевірка на існування  користувача
            if (dbProfile == null)
                throw new UserNotFoundException("Користувача не знайдено");

            var profile = Mapper.Map<User>(dbProfile);
            return profile;
        }

        public GetUserProfileQueryHandler(AppDbContext dbContext, IMapper mapper)
            : base(dbContext, mapper)
        {
        }
    }
}
```

Файл GetUserRequestsQuery.cs

```
using System.Runtime.InteropServices.JavaScript;
using Domain.Models;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.Extensions.Logging;
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter.Helpers;
```

```
using Infra.DatabaseAdapter.Models;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class GetUserRequestsQuery : IRequest<List<LessonRequest>>
{
   public int? UserId { get; set; }

   public bool IsTutor { get; set; }

   public class GetUserRequestsQueryHandler : BaseMediatrHandler<GetUserRequestsQuery, List<LessonRequest>>
   {
      public override async Task<List<LessonRequest>> Handle(GetUserRequestsQuery r, CancellationToken token)
      {
         List<LessonRequest> listRequest = new();
         var q = DatabaseContext.Requests.AsNoTracking()
            .Include(x => x.Tutor)
            .ThenInclude(x => x.User)
            .Include(x => x.Subject)
            .Include(x => x.Created).AsQueryable();

         if (r.IsTutor)
            q = q.Where(x => x.TutorId == r.UserId);
         else
            q = q.Where(x => x.CreatedId == r.UserId);

         q = q.OrderBy(x => x.Status); // Перші ті, що не оброблені

         foreach (var dbRequest in await q.ToArrayAsync())
         {
            var req = Mapper.Map<LessonRequest>(dbRequest);
            req.TutorName = dbRequest.Tutor.User.FullName();
            req.Subject = dbRequest.Subject.Name;
            req.IsTutor = r.IsTutor;
            req.UserName = dbRequest.Created.FullName();
            listRequest.Add(req);
         }

         return listRequest;
      }

      public GetUserRequestsQueryHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
      {
      }
   }
}
```

Файл UserIsTutorQuery.cs

```
using AutoMapper;
using Domain.Helpers;
using Infra.DatabaseAdapter;
using MediatR;
using Microsoft.EntityFrameworkCore;

namespace Domain.Queries;

public class UserIsTutorQuery : IRequest<bool>
{
   public int UserId { get; set; }

   public class UserIsTutorQueryHandler : BaseMediatrHandler<UserIsTutorQuery, bool>
   {
      public UserIsTutorQueryHandler(AppDbContext dbContext, IMapper mapper)
         : base(dbContext, mapper)
```

```
        {
        }


        public override async Task<bool> Handle(UserIsTutorQuery r, CancellationToken token)
        {
            //Перевірка на існування  користувача
            var dbProfile = await DatabaseContext.Users.AsNoTracking().FirstOrDefaultAsync(x => x.Id == r.UserId);
            if (dbProfile == null)
                throw new UserNotFoundException("Користувача не знайдено");
            return dbProfile.ProfileEnabled;
        }
    }
}
```

Файл CourseRequestStatus.cs

```
using System.ComponentModel.DataAnnotations;

namespace Infra.DatabaseAdapter.Helpers;

public enum LessonRequestStatus
{
    [Display(Name = "Новий")] New,
    [Display(Name = "Схвалено")] Approved,
    [Display(Name = "Відхилено")] Rejected
}
```

Файл ITrackable.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.ChangeTracking;

namespace Infra.DatabaseAdapter.Helpers;

public interface ITrackable
{
    DateTime CreatedAt { get; set; }
    DateTime? UpdatedAt { get; set; }

    static void BeforeSaving(IEnumerable<EntityEntry> entries)
    {
        var now = DateTime.UtcNow;
        foreach (var entry in entries)
            if (entry.Entity is ITrackable trackable)
                switch (entry.State)
                {
                    case EntityState.Modified:
                        trackable.UpdatedAt = now;
                        entry.Property("CreatedAt").IsModified = false;
                        break;
                    case EntityState.Added:
                        trackable.CreatedAt = now;
                        trackable.UpdatedAt = now;
                        break;
                }
    }
}
```

Файл SolutionStatus.cs

```
using System.ComponentModel.DataAnnotations;

namespace Infra.DatabaseAdapter.Helpers;

public enum SolutionStatus
{
    [Display(Name = "До виконання")] Todo,
```

```
    [Display(Name = "На Перевирці")] Review,
    [Display(Name = "Виконано")] Completed
}
```

Файл TimeOnlyConverter.cs

```
using Microsoft.EntityFrameworkCore.ChangeTracking;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace Infra.DatabaseAdapter.Helpers;

public class TimeOnlyConverter : ValueConverter<TimeOnly, TimeSpan>
{
    public TimeOnlyConverter() : base(
        timeOnly => timeOnly.ToTimeSpan(),
        timeSpan => TimeOnly.FromTimeSpan(timeSpan))
    {
    }
}
```

Файл AboutTutorModel.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Xml;

namespace Infra.DatabaseAdapter.Models;

[Table("AboutTutor")]
public class AboutTutorModel
{
    [Required] public int Id { get; set; }
    [MaxLength(5000)] public string Content { get; set; } = string.Empty;
}
```

Файл AssignmentModel.cs

```
using System.ComponentModel.DataAnnotations;
using Infra.DatabaseAdapter.Helpers;

namespace Infra.DatabaseAdapter.Models;

public class AssignmentModel : ITrackable
{
    public int Id { get; set; }
    public int TutorId { get; set; }
    public TutorModel Tutor { get; set; } = null!;

    public int SubjectId { get; set; }
    public SubjectModel Subject { get; set; } = null!;
    [MaxLength(50)] public string Title { get; set; } = string.Empty;
    [MaxLength(500)] public string Description { get; set; } = string.Empty;
    public DateTime Deadline { get; set; }

    public virtual List<UserFileModel> Files { get; set; } = [];
    public virtual List<SolutionModel> Solutions { get; set; } = [];

    //ITrackable
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }
}
```

Файл AvailableTime.cs

```
using Infra.DatabaseAdapter.Helpers;

namespace Infra.DatabaseAdapter.Models;
```

```
public class AvailableTimeModel : ITrackable, ICloneable
{
   public int Id { get; set; }
   public int DayOfWeek { get; set; }
   public TimeOnly StartTime { get; set; }
   public TimeOnly EndTime { get; set; }
   public int ProfileId { get; set; }
   public TutorModel Profile { get; set; } = null!;

   //ITrackable
   public DateTime CreatedAt { get; set; }
   public int CreatedId { get; set; }
   public DateTime? UpdatedAt { get; set; }
   public int? UpdatedId { get; set; }
   public object Clone() => MemberwiseClone();
}
```

Файл CityModel.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using Infra.DatabaseAdapter.Helpers;

namespace Infra.DatabaseAdapter.Models;

public class CityModel : ITrackable
{
   public int Id { get; set; }

   [DisplayName("Назва")]
   [MaxLength(100)]
   [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
   public string Name { get; set; } = string.Empty;

   [MaxLength(140)]
   [DisplayName("Область")]
   [Required(ErrorMessage = "Поле '{0}' є обов'язковим")]
   public string Region { get; set; } = string.Empty;

   //ITrackable
   [DisplayName("Додано")]
   [DisplayFormat(DataFormatString = "{0:d}")]
   public DateTime CreatedAt { get; set; }

   [DisplayFormat(DataFormatString = "{0:d}")]
   [DisplayName("Оновлено в")]
   public DateTime? UpdatedAt { get; set; }


   public string FullName() => $"{Name},{Region}";
}
```

Файл FileModel.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using Infra.DatabaseAdapter.Helpers;

namespace Infra.DatabaseAdapter.Models;

public class UserFileModel : ITrackable
{
   public Guid Id { get; set; }
   [MaxLength(100)] public string FileName { get; set; } = string.Empty;
   [MaxLength(350)] public string ServerName { get; set; } = string.Empty;

   public int OwnerId { get; set; }
```

```
    public UserModel Owner { get; set; } = null!;

    //ITrackable
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }

    public virtual List<AssignmentModel> Assignments { get; set; } = [];
    public virtual List<SolutionModel> Solutions { get; set; } = [];
}
```

Файл LessonModel.cs

```
using System.ComponentModel.DataAnnotations;
using Infra.DatabaseAdapter.Helpers;

namespace Infra.DatabaseAdapter.Models;

public class LessonModel : ITrackable
{
    public int Id { get; set; }
    public int TutorId { get; set; }
    public TutorModel Tutor { get; set; } = null!;
    public DateTime From { get; set; }
    public DateTime To { get; set; }
    [MaxLength(50)] public string Title { get; set; } = string.Empty;
    [MaxLength(200)] public string Comment { get; set; } = string.Empty;

    public int? RequestId { get; set; }
    public RequestModel? Request { get; set; } = null;
    public int SubjectId { get; set; }
    public SubjectModel Subject { get; set; } = null!;

    public virtual List<UserModel> Students { get; set; } = [];


    //ITrackable
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }
}
```

Файл RequestModel.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using Infra.DatabaseAdapter.Helpers;

namespace Infra.DatabaseAdapter.Models;

public class RequestModel : ITrackable
{
    public int Id { get; set; }
    public int TutorId { get; set; }
    public TutorModel Tutor { get; set; } = null!;
    public LessonRequestStatus Status { get; set; }
    public DateTime From { get; set; }
    public DateTime To { get; set; }
    [MaxLength(300)] public string Comment { get; set; } = string.Empty;
    [MaxLength(300)] public string TutorComment { get; set; } = string.Empty;

    public int SubjectId { get; set; }
    public SubjectModel Subject { get; set; } = null!;
    public int CreatedId { get; set; }
    public UserModel Created { get; set; } = null!;


    //ITrackable
    public DateTime CreatedAt { get; set; }
```

```
    public DateTime? UpdatedAt { get; set; }
}
```

Файл ReviewModel.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using Infra.DatabaseAdapter.Helpers;
using Microsoft.EntityFrameworkCore;

namespace Infra.DatabaseAdapter.Models;

public class ReviewModel : ITrackable
{
    public int Id { get; set; }
    public int TutorId { get; set; }
    public TutorModel Tutor { get; set; } = null!;
    [Range(0, 10)] public int Rating { get; set; }
    [MaxLength(1000)] public string Description { get; set; } = string.Empty;

    public int AuthorId { get; set; }
    public UserModel Author { get; set; } = null!;

    //ITrackable
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }
}
```

Файл SolutionModel.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using Infra.DatabaseAdapter.Helpers;

namespace Infra.DatabaseAdapter.Models;

public class SolutionModel : ITrackable
{
    public int Id { get; set; }
    public int AssignmentId { get; set; }
    public AssignmentModel Assignment { get; set; } = null!;

    public int StudentId { get; set; }
    public UserModel Student { get; set; } = null!;

    public SolutionStatus Status { get; set; }

    [MaxLength(500)] public string Answer { get; set; } = string.Empty;
    [MaxLength(500)] public string TutorComment { get; set; } = string.Empty;
    public virtual List<UserFileModel> Files { get; set; } = [];

    //ITrackable
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }
}
```

Файл SubjectModel.cs

```
using System.ComponentModel.DataAnnotations;

namespace Infra.DatabaseAdapter.Models;

public class SubjectModel
{
    public int Id { get; set; }
    [MaxLength(100)] public string Name { get; set; } = string.Empty;
```

```
    public virtual List<TutorModel> Profiles { get; set; } = [];
}
```

Файл TutorModel.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using Infra.DatabaseAdapter.Helpers;

namespace Infra.DatabaseAdapter.Models;

public class TutorModel : ITrackable
{
    public int Id { get; set; }

    public UserModel User { get; set; } = null!;

    [Required] public AboutTutorModel About { get; set; } = new();
    [MaxLength(300)] public string Address { get; set; } = string.Empty;

    public bool OnlineAccess { get; set; }
    public bool TutorHomeAccess { get; set; }
    public bool StudentHomeAccess { get; set; }


    [MaxLength(300)] public string Descriptions { get; set; } = string.Empty;


    [DataType(DataType.Currency)]
    [Range(10, 9999)]
    public int HourRate { get; set; }

    public virtual List<SubjectModel> Subjects { get; set; } = [];
    public virtual List<AvailableTimeModel> AvailableTimes { get; set; } = [];
    public virtual List<RequestModel> Requests { get; set; } = [];
    public virtual List<LessonModel> TeachingLessons { get; set; } = [];

    public virtual List<ReviewModel> Reviews { get; set; } = [];
    public virtual List<UserModel> InFavorite { get; set; } = [];

    //ITrackable
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }
}
```

Файл UserModel.cs

```
using System.ComponentModel.DataAnnotations;
using System.Net.Mail;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.ChangeTracking;

namespace Infra.DatabaseAdapter.Models;

public class UserModel : IdentityUser<int>
{
    [MaxLength(50)] public string Name { get; set; } = string.Empty;
    [MaxLength(50)] public string Surname { get; set; } = string.Empty;
    [MaxLength(50)] public string Patronymic { get; set; } = string.Empty;
    public bool ProfileEnabled { get; set; }
    public TutorModel Tutor { get; set; } = null!;

    public int? CityId { get; set; }
    public CityModel? City { get; set; }
    public DateTime? BirthDate { get; set; }
```

```
    public virtual List<TutorModel> FavoriteTutors { get; set; } = [];
    public virtual List<LessonModel> Lessons { get; set; } = [];
    public virtual List<RequestModel> Requests { get; set; } = [];
    public virtual List<SolutionModel> Solutions { get; set; } = [];
    public virtual List<ReviewModel> Reviews { get; set; } = [];

    public virtual List<UserFileModel> Files { get; set; } = [];

    public string FullName() => Name.Length > 2 ? $"{Name} {Patronymic} {Surname}" : Email ?? "";

    public string NormalizeName { get; set; } = string.Empty;

    public static void BeforeSaving(IEnumerable<EntityEntry> entries)
    {
        foreach (var entry in entries)
            if (entry.Entity is UserModel u)
                if (entry.State is EntityState.Added or EntityState.Modified)
                {
                    if (u.Name.Length > 2)
                        u.NormalizeName = string.Join(' ', u.Name, u.Patronymic, u.Surname);
                    else
                        u.NormalizeName = u.Email ?? "";

                    u.NormalizeName = u.NormalizeName.ToUpperInvariant();
                }
    }
}
```

Файл AppDbContext.cs

```
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using Org.BouncyCastle.Ocsp;
using SixLabors.ImageSharp.Processing.Processors.Convolution;

namespace Infra.DatabaseAdapter;

public class AppDbContext : IdentityDbContext<UserModel, IdentityRole<int>, int>
{
    private bool _withSeed;

    public AppDbContext(DbContextOptions<AppDbContext> options, bool withSeed = false) : base(options) =>
        _withSeed = withSeed;

    public DbSet<CityModel> Cities { get; set; } = null!;
    public DbSet<SubjectModel> Subjects { get; set; } = null!;
    public DbSet<TutorModel> Tutors { get; set; } = null!;
    public DbSet<AvailableTimeModel> AvailableTimes { get; set; } = null!;
    public DbSet<LessonModel> Lessons { get; set; } = null!;
    public DbSet<AssignmentModel> Assignments { get; set; } = null!;
    public DbSet<SolutionModel> Solutions { get; set; } = null!;
    public DbSet<UserFileModel> Files { get; set; } = null!;
    public DbSet<RequestModel> Requests { get; set; } = null!;
    public DbSet<ReviewModel> Reviews { get; set; } = null!;



    private void InitialSeed(ModelBuilder builder)
    {
        builder.Entity<IdentityRole<int>>().HasData(DataSeed.Roles);
        builder.Entity<SubjectModel>().HasData(DataSeed.Subjects);
        builder.Entity<CityModel>().HasData(DataSeed.Cities);
    }
```

```
protected override void OnModelCreating(ModelBuilder builder)
{
    builder.Entity<UserModel>(entity =>
    {
        entity.HasOne(x => x.Tutor)
            .WithOne(x => x.User)
            .HasForeignKey<TutorModel>(x => x.Id);
        entity.Navigation(x => x.Tutor)
            .IsRequired();
        entity.HasMany(x => x.FavoriteTutors)
            .WithMany(x => x.InFavorite)
            .UsingEntity("Favorites");
    });

    builder.Entity<TutorModel>(entity =>
    {
        entity.HasOne(x => x.About).WithOne()
            .HasForeignKey<AboutTutorModel>(x => x.Id);
        entity.Navigation(x => x.About).IsRequired();
    });

    builder.Entity<LessonModel>()
        .HasMany(x => x.Students)
        .WithMany(x => x.Lessons)
        .UsingEntity("StudentLessons");

    builder.Entity<TutorModel>()
        .HasMany(x => x.Subjects)
        .WithMany(x => x.Profiles)
        .UsingEntity("ProfileSubjects");

    builder.Entity<AssignmentModel>()
        .HasMany(x => x.Files)
        .WithMany(x => x.Assignments)
        .UsingEntity("AssignmentFiles");

    builder.Entity<SolutionModel>()
        .HasMany(x => x.Files)
        .WithMany(x => x.Solutions)
        .UsingEntity("SolutiontFiles");

    builder.Entity<AvailableTimeModel>()
        .Property(x => x.StartTime)
        .HasConversion(new TimeOnlyConverter());
    builder.Entity<AvailableTimeModel>()
        .Property(x => x.EndTime)
        .HasConversion(new TimeOnlyConverter());

    //Seed
    if (_withSeed)
        InitialSeed(builder);
    base.OnModelCreating(builder);
}


public override int SaveChanges(bool acceptAllChangesOnSuccess)
{
    var entries = ChangeTracker.Entries()
        .Where(x => x.State is EntityState.Added or EntityState.Modified)
        .ToList();
    ITrackable.BeforeSaving(entries);
    UserModel.BeforeSaving(entries);

    return base.SaveChanges(acceptAllChangesOnSuccess);
}

public override Task<int> SaveChangesAsync(bool acceptAllChangesOnSuccess,
```

```
        CancellationToken cancellationToken = new())
    {
        var entries = ChangeTracker.Entries()
            .Where(x => x.State is EntityState.Added or EntityState.Modified)
            .ToList();
        ITrackable.BeforeSaving(entries);
        UserModel.BeforeSaving(entries);
        return base.SaveChangesAsync(acceptAllChangesOnSuccess, cancellationToken);
    }
}
```

Файл DataSeed.cs

```
using Infra.DatabaseAdapter.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;

namespace Infra.DatabaseAdapter;

public class DataSeed
{
    public static SubjectModel[] Subjects
    {
        get
        {
            var iSubject = 1;
            var objSubject = File.ReadAllLines("../Infra/Data/subjects_list.csv")
                .Select(line => line.Split(','))
                .Select(x => new SubjectModel
                {
                    Id = iSubject++,
                    Name = x[0]
                })
                .ToArray();
            return objSubject;
        }
    }

    public static CityModel[] Cities
    {
        get
        {
            var iCity = 1;
            var objCity = File.ReadAllLines("../Infra/Data/ukr_cities.csv")
                .Select(line => line.Split(','))
                .Select(x => new CityModel
                {
                    Id = iCity++,
                    Name = x[0],
                    Region = x[1],
                    CreatedAt = DateTime.Parse("2024-01-01")
                })
                .ToArray();
            return objCity;
        }
    }

    public static IdentityRole<int>[] Roles =>
    [
        new IdentityRole<int>
        {
            Id = 1,
            Name = "Administrator",
            NormalizedName = "ADMINISTRATOR"
        },
        new IdentityRole<int>
        {
```

```
            Id = 2,
            Name = "User",
            NormalizedName = "USER"
        },
        new IdentityRole<int>
        {
            Id = 3,
            Name = "Removed",
            NormalizedName = "REMOVED"
        }
    ];
}
```

Файл IStorageRepository.cs

```
namespace Infra.Storage;

public interface IStorageRepository
{
    public void RemoveFileIfExist(string filename);
    public StorageFile SaveFile(string extension, byte[] data);
    public void SaveAvatar(string filename, byte[] data);
    public StorageFile GetFile(string serverName);
}
```

Файл StorageFile.cs

```
namespace Infra.Storage;

public class StorageFile
{
    public readonly MemoryStream Memory;
    public readonly string Filename;
    public Guid Guid;

    public StorageFile(MemoryStream memory, string filename, Guid? guid = null)
    {
        Memory = memory;
        Filename = filename;
        if (guid.HasValue)
            Guid = guid.Value;
    }
}
```

Файл StorageRepository.cs

```
using System.Configuration;
using Microsoft.Extensions.Configuration;
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.Formats.Png;
using SixLabors.ImageSharp.Processing;

namespace Infra.Storage;

public class StorageRepository : IStorageRepository
{
    private readonly string _filesFolder;
    private readonly string _avatarFolder;

    public StorageRepository(IConfiguration configuration)
    {
        _filesFolder = configuration["FilesFolder"] ?? string.Empty;
        _avatarFolder = configuration["AvatarFolder"] ?? string.Empty;
    }


    public void CheckFolder(string filesFolder)
    {
```

```
        if (filesFolder == string.Empty || filesFolder.Length == 0)
            throw new ConfigurationErrorsException("The 'FilesFolder' key does not exist in appsettings.json");
        else if (!Directory.Exists(filesFolder))
            throw new DirectoryNotFoundException($"FilesFolder: Directory {filesFolder} does not exist");
    }

    public void RemoveFileIfExist(string filename)
    {
        CheckFolder(_filesFolder);

        // Перевірка, чи існує файл із повним шляхомкщо файл знайдено, видаліть його
        if (File.Exists(Path.Combine(_filesFolder, filename))) File.Delete(Path.Combine(_filesFolder, filename));
    }

    /// <summary>
    /// Return guid Id and filename for saved file
    /// </summary>
    /// <param name="extension">.pdf</param>
    /// <param name="data"></param>
    /// <returns></returns>
    public StorageFile SaveFile(string extension, byte[] data)
    {
        CheckFolder(_filesFolder);

        if (extension.Length > 0 && extension[0] != '.')
            extension = "." + extension;

        //Згенерувати шлях до файлу
        var guid = Guid.NewGuid();
        var serverFile = new StorageFile(new MemoryStream(data), guid.ToString() + extension, guid);
        var fullPath = Path.Combine(_filesFolder, serverFile.Filename);

        //Змінити та зберегти в папку з файлами
        using (var fs = new FileStream(fullPath, FileMode.Create))
            fs.Write(data, 0, data.Length);

        return serverFile;
    }

    public void SaveAvatar(string userFile, byte[] data)
    {
        CheckFolder(_avatarFolder);

        using (var fs = new FileStream(Path.Combine(_avatarFolder, userFile + ".png"), FileMode.Create))
        using (var image = Image.Load(data))
        {
            image.Mutate(x => x.Resize(200, 200));
            image.Save(fs, new PngEncoder());
        }
    }


    public StorageFile GetFile(string serverName)
    {
        // Перевірка, чи існує файл
        var path = Path.Combine(_filesFolder, serverName);
        if (!File.Exists(path))
            throw new FileNotFoundException("File not found in storage");

        //Копирование файла в память
        var memory = new MemoryStream();
        using (var stream = new FileStream(path, FileMode.Open)) stream.CopyTo(memory);
        memory.Position = 0;
        return new StorageFile(memory, serverName);
    }
}
```

Файл avatar.js

```
const avatar_path = "/avatars/default.png";

var elements = document.getElementsByClassName('img-load-control');
for (var i = 0; i < elements.length; ++i) {
    elements[i].addEventListener("error", function () {
        // console.log(this)
        this.src = avatar_path
    })
    if (typeof elements[i].naturalWidth != "undefined" && elements[i].naturalWidth === 0) {
        elements[i].src = avatar_path;
    }
}
```

Файл details-calendar.js

```
createCalendar()

function createCalendar() {
    calElement = document.getElementById('calendar')
    calendar = new FullCalendar.Calendar(calElement, {
        initialDate: Date.now(),
        initialView: 'timeGridWeek',
        themeSystem: 'bootstrap4',
        headerToolbar: {
            left: '',
            center: '',
            right: ''
        },
        selectOverlap: false,
        eventOverlap: false,
        height: "auto",
        navLinks: false,
        editable: true,
        selectable: true,
        selectMirror: true,
        allDaySlot: false,
        firstDay: 1,
        select: addNewEvent,
        eventSources: getEvents
    });
    calendar.render();
}

//Отримання списку подій
function getEvents(info, successCallback, failureCallback) {
    var strStart = new Date(info.start - info.start.getTimezoneOffset() * 60000).toISOString()
    var strEnd = new Date(info.end - info.end.getTimezoneOffset() * 60000).toISOString()

    $.ajax({
        url: "/Session/List",
        type: "GET",
        dataType: 'json',
        data: {
            from: strStart,
            to: strEnd,
            userid: $("#calendar").data('tutorid')
        },
        success: function (data) {
            availableDates = data.filter(date => date.type.toLowerCase() !== "available");
            calEvents = availableDates.map(({from, to}) => ({
                groupId: "busy",
                start: from,
                end: to,
                display: 'background',
```

```
            backgroundColor: 'red'
        }));
        successCallback(calEvents)
    }
  })
}

//Додання нового запису
function addNewEvent(info) {
  var userid = $("#calendar").data('userid')
  if (userid > 0) {
    //'2024-04-23T05:30:00+03:00' --> '2024-04-23T05:30:00'
    if (info != null) {
      document.requestForm.timeFrom.value = info.startStr.slice(0, -6)
      document.requestForm.timeTo.value = info.endStr.slice(0, -6)
    }

    $('#requestModal').modal('show');
  } else {
    $('#auth-modal').modal('show');
  }
}


//зберегти в базу
async function send_request() {
  formData = {
    TutorId: document.requestForm.tutorProfileId.value,
    SubjectId: document.requestForm.subject.value,
    From: document.requestForm.timeFrom.value,
    To: document.requestForm.timeTo.value,
    Comment: document.requestForm.tutorComment.value ?? ""
  }
  console.log(formData)
  $.ajax({
    type: "POST",
    url: document.requestForm.action,
    data: formData,
    success: function () {
      $('#requestModal').modal('hide');
    }, error: function (data) {
      alert(data.responseJSON.join('\n')); // show response from the php script.
    }
  })
}
```

Файл edit-calendar.js

```
createCalendar()

function createCalendar() {
  calElement = document.getElementById('calendar')
  calendar = new FullCalendar.Calendar(calElement, {
    initialDate: Date.now(),
    initialView: 'timeGridWeek',
    themeSystem: 'bootstrap4',
    headerToolbar: {
      left: '',
      center: '',
      right: ''
    },
    selectOverlap: false,
    eventOverlap: false,
    height: "auto",
    navLinks: false,
    editable: false,
    selectable: true,
```

```
        selectMirror: true,
        allDaySlot: false,
        firstDay: 1,
        select: addNewEvent,
        eventClick: removeEvent,
        eventSources: [{events: getEvents}]
    });
    calendar.render();
}


//Отримання списку подій
function getEvents(info, successCallback, failureCallback) {
    var strStart = new Date(info.start - info.start.getTimezoneOffset() * 60000).toISOString()
    var strEnd = new Date(info.end - info.end.getTimezoneOffset() * 60000).toISOString()
    $.ajax({
        url: "/Session/List",
        type: "GET",
        dataType: 'json',
        data: {
            from: strStart,
            to: strEnd,
            userid: $("#calendar").data('tutorid'),
            TimeTypes: "Available"
        },
        success: function (data) {
            calEvents = data.map((x) => ({
                start: x.from,
                end: x.to,
                id: x.id,
                title: x.title
            }));
            successCallback(calEvents)
        }
    })
}


function addNewEvent(info) {
    var endTimeDiff = 0;
    if (info.start.getDate() !== info.end.getDate())
        endTimeDiff = 1;
    var endTime = new Date(info.end - endTimeDiff - info.start.getTimezoneOffset() * 60000).toISOString()
    $.ajax({
        url: "/Session/Create",
        type: "POST",
        dataType: 'json',
        async: false,
        data: {
            Type: "Available",
            from: info.startStr,
            to: endTime.toString(),
            userid: $("#calendar").data('tutorid'),
            TimeTypes: "Available"
        },
        success: function (data) {
            calendar.refetchEvents();
        },
        error: function (data) {
            alert("Помилка")
            calendar.refetchEvents();
        }
    })
}

function removeEvent(info) {
    $.ajax({
```

```
        url: "/Session/Delete",
        type: "POST",
        dataType: 'json',
        data: {
            Id: info.event.id,
        },
        success: function (data) {
            calendar.refetchEvents();
        },
        error: function (data) {
            alert("Помилка")
            calendar.refetchEvents();
        }
    })
}
```

Файл file.js

```
const alertPlaceholder = document.getElementById('liveAlertPlaceholder')
const appendAlert = (message, type) => {
    const wrapper = document.createElement('div')
    wrapper.innerHTML = [
        `<div class="alert alert-${type} alert-dismissible" role="alert">`,
        `   <div>${message}</div>`,
        '   <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>',
        '</div>'
    ].join('')
    alertPlaceholder.append(wrapper)
}

var uploadForm = document.getElementById('uploadform');
if (uploadForm !== null && uploadForm !== undefined)
    document.getElementById('uploadform-file').onchange = function () {
        $(".alert").alert('close');
        var postData = new FormData($(this.parentElement)[0]);
        $.ajax({
            url: this.parentElement.action,
            type: "POST",
            data: postData,
            processData: false,
            contentType: false,
            success: function (data, text) {
                appendAlert('Файл успішно завантажено на  сервер', 'success')
            },
            error: function (request, status, error) {
                var msg = "Помилка під час завантаження файлу, спробуйте надіслати інший файл"
                if (request !== undefined) {
                    if (request.responseJSON !== undefined && request.responseJSON.message !== undefined)
                        msg += ": " + request.responseJSON.message
                    else (request.responseText !== undefined)
                    msg += ": " + request.responseText

                }

                appendAlert(msg + "\n", 'danger')
            }
        });
    };
```

Файл main.js

```
let table = new DataTable('.pretty-table', {
    language: {
        "lengthMenu": "Відображати _MENU_ записів на сторінку",
        "zeroRecords": "Нічого не знайдено - вибачте",
        "info": "Показано сторінку _PAGE_ з _PAGES_",
        "infoEmpty": "Немає доступних записів",
```

```
        "infoFiltered": "(відфільтровано з _MAX_ загальних записів)",
        "emptyTable": "Немає даних у таблиці",
        "loadingRecords": "Завантаження...",
        "search": "Пошук:",
        "арія": {
            "orderable": "Упорядкувати за цим стовпцем",
            "orderableReverse": "Змінити порядок цього стовпця"
        }
    },
    pagingType: 'simple_numbers',
    scrollCollapse: true,
    deferRender: true,
    scroller: true,
    scrollY: 1000,

});


//load status favorite button
function FavoriteStatus(element, isGet) {
    var urlMethod = isGet ? 'get' : 'post';
    $.ajax({
        url: "/Profile/IsFavorite",
        method: urlMethod,
        dataType: 'json',
        data: {
            UserId: element.dataset.userid,
            TutorId: element.dataset.tutorid,
            Status: element.checked
        },
        success: function (data) {
            element.checked = data.status;
        },
        error: function (r, t, e) {
            element.checked = !element.checked //revert state
            console.log(e)
        }
    });
}


const favoriteElem = document.getElementById("btn-favorite")
if (favoriteElem !== null) {
    FavoriteStatus(favoriteElem, isGet = true)
//onclick for favorite button

    favoriteElem.addEventListener("click", function () {
        FavoriteStatus(this)
    })

}
```

Файл sb-admin-2.js

```
(function($) {
  "use strict"; // Start of use strict

  // Toggle the side navigation
  $("#sidebarToggle, #sidebarToggleTop").on('click', function(e) {
    $("body").toggleClass("sidebar-toggled");
    $(".sidebar").toggleClass("toggled");
    if ($(".sidebar").hasClass("toggled")) {
      $('.sidebar .collapse').collapse('hide');
    };
  });

  // Close any open menu accordions when window is resized below 768px
  $(window).resize(function() {
```

```
  if ($(window).width() < 768) {
    $('.sidebar .collapse').collapse('hide');
  };

  // Toggle the side navigation when window is resized below 480px
  if ($(window).width() < 480 && !$(".sidebar").hasClass("toggled")) {
    $("body").addClass("sidebar-toggled");
    $(".sidebar").addClass("toggled");
    $('.sidebar .collapse').collapse('hide');
  };
});

// Prevent the content wrapper from scrolling when the fixed side navigation hovered over
$('body.fixed-nav .sidebar').on('mousewheel DOMMouseScroll wheel', function(e) {
  if ($(window).width() > 768) {
    var e0 = e.originalEvent,
      delta = e0.wheelDelta || -e0.detail;
    this.scrollTop += (delta < 0 ? 1 : -1) * 30;
    e.preventDefault();
  }
});

// Scroll to top button appear
$(document).on('scroll', function() {
  var scrollDistance = $(this).scrollTop();
  if (scrollDistance > 100) {
    $('.scroll-to-top').fadeIn();
  } else {
    $('.scroll-to-top').fadeOut();
  }
});

// Smooth scrolling using jQuery easing
$(document).on('click', 'a.scroll-to-top', function(e) {
  var $anchor = $(this);
  $('html, body').stop().animate({
    scrollTop: ($($anchor.attr('href')).offset().top)
  }, 1000, 'easeInOutExpo');
  e.preventDefault();
});

})(jQuery); // End of use strict
```

Файл CityController.cs

```
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace Web.Controllers.Admin;

[Authorize(Roles = "Administrator")]
[Route("/[controller]/[action]")]
[Produces("text/html; charset=utf-8")]
public class CityController : Controller
{
    private readonly AppDbContext _context;

    public CityController(AppDbContext context) => _context = context;

    // GET: City
    [HttpGet]
    public async Task<IActionResult> Index() => View(await _context.Cities.ToListAsync());


    // GET: City/Create
```

```
[HttpGet]
public IActionResult Create() => View();

// POST: City/Create
[HttpPost]
public async Task<IActionResult> Create(
    [Bind("Id,Name,Region,CreatedAt,UpdatedAt")]
    CityModel cityModel)
{
    ModelState.Remove("Id");
    if (ModelState.IsValid)
    {
        _context.Add(cityModel);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    return View(cityModel);
}

// GET: City/Edit/5
[HttpGet]
[Route("{id?}")]
public async Task<IActionResult> Edit(int? id)
{
    if (id == null) return NotFound();

    var cityModel = await _context.Cities.FindAsync(id);
    if (cityModel == null) return NotFound();
    return View(cityModel);
}

// POST: City/Edit/5
[HttpPost]
[Route("{id}")]
public async Task<IActionResult> Edit(int id,
    [Bind("Id,Name,Region,CreatedAt,UpdatedAt")]
    CityModel cityModel)
{
    if (id != cityModel.Id) return NotFound();

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(cityModel);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!CityModelExists(cityModel.Id))
                return NotFound();
            else
                throw;
        }

        return RedirectToAction(nameof(Index));
    }

    return View(cityModel);
}

// GET: City/Delete/5
[HttpGet]
[Route("{id?}")]
public async Task<IActionResult> Delete(int? id)
{
```

```
        if (id == null) return NotFound();

        var cityModel = await _context.Cities
            .FirstOrDefaultAsync(m => m.Id == id);
        if (cityModel == null) return NotFound();

        return View(cityModel);
    }

    // POST: City/Delete/5
    [HttpPost]
    [Route("{id}")]
    [ActionName("Delete")]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        var cityModel = await _context.Cities.FindAsync(id);
        if (cityModel != null) _context.Cities.Remove(cityModel);

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool CityModelExists(int id) => _context.Cities.Any(e => e.Id == id);
}
```

Файл RemoveUserController.cs

```
using System.Security.Claims;
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Helpers;
using Infra.DatabaseAdapter.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Web.Models.RemoveUser;

namespace Web.Controllers.Admin;

public class RemoveUserController : Controller
{
    private int IdentityId => Convert.ToInt32(User.FindFirstValue(ClaimTypes.NameIdentifier));
    private readonly AppDbContext _context;
    private readonly SignInManager<UserModel> _signInManager;
    private readonly UserManager<UserModel> _userManager;


    public RemoveUserController(UserManager<UserModel> userManager, SignInManager<UserModel> signInManager,
        AppDbContext context)
    {
        _context = context;
        _signInManager = signInManager;
        _userManager = userManager;
    }

    public int? RemoveRoleId => _context.Roles
        .AsNoTracking()
        .Where(x => x.NormalizedName == "Removed")
        .FirstOrDefault()?.Id ?? null;

    [Authorize(Roles = "Administrator")]
    public async Task<IActionResult> Index()
    {
        var model = new RemoveUserVm();
        //get roles
```

```
    if (RemoveRoleId != null)
    {
        var userList = _context.Users
            .AsNoTracking()
            .Where(x => x.Id != IdentityId)
            .Select(x => new SelectListItem { Value = x.Id.ToString(), Text = $"{x.Id},{x.NormalizeName}" })
            .ToList();

        var removedUsers = await _context.UserRoles
            .AsNoTracking()
            .Where(x => x.RoleId == RemoveRoleId)
            .Select(x => x.UserId)
            .ToListAsync();

        //ignore removed users
        foreach (var item in userList)
            if (!removedUsers.Contains(int.Parse(item.Value)))
                model.UserList.Add(item);
    }

    return View(model);
}

[Authorize(Roles = "Removed")]
public IActionResult Removed() => View();


[HttpPost]
[Authorize(Roles = "Administrator")]
public async Task<IActionResult> DeactivateUser(int userId)
{
    var userDb = await _context.Users
        .Where(x => x.Id == userId)
        .FirstOrDefaultAsync();

    if (RemoveRoleId != null && userDb != null) //add role Removed
    {
        //remove role from db
        _context.UserRoles.Add(new IdentityUserRole<int>() { UserId = userDb.Id, RoleId = RemoveRoleId.Value });
        await _context.SaveChangesAsync();

        // reset user cookies
        await _userManager.UpdateSecurityStampAsync(userDb);
    }


    return Redirect(nameof(Index));
}

[HttpPost]
[Authorize(Roles = "Removed")]
public async Task<IActionResult> RemoveMePersonalData()
{
    var userDb = await _context.Users
        .Include(x => x.Tutor)
        .Include(x => x.Tutor.About)
        .Include(x => x.Tutor.AvailableTimes)
        .Where(x => x.Id == IdentityId)
        .FirstAsync();


    //remove all roles
    var rolesList = _context.UserRoles.Where(x => x.UserId == IdentityId);
    _context.UserRoles.RemoveRange(rolesList);

    //clear user fields
    userDb.Name = string.Empty;
```

```
        userDb.Surname = string.Empty;
        userDb.Patronymic = string.Empty;
        userDb.ProfileEnabled = false;
        userDb.CityId = null;
        userDb.BirthDate = null;
        userDb.FavoriteTutors = new List<TutorModel>();
        userDb.NormalizeName = string.Empty;
        userDb.UserName = null;
        userDb.Email = null;
        userDb.NormalizedEmail = null;
        userDb.NormalizedUserName = null;
        userDb.PasswordHash = null;

        //clear tutor fields
        if (userDb.Tutor != null)
        {
            userDb.Tutor.Address = string.Empty;
            userDb.Tutor.OnlineAccess = false;
            userDb.Tutor.TutorHomeAccess = false;
            userDb.Tutor.StudentHomeAccess = false;
            userDb.Tutor.Descriptions = string.Empty;
            userDb.Tutor.HourRate = 0;
            userDb.Tutor.Subjects = new List<SubjectModel>();

            //remove About
            if (userDb.Tutor.About.Id != 0)
                _context.Remove(userDb.Tutor.About);

            //remove AvailableTimes
            _context.RemoveRange(userDb.Tutor.AvailableTimes);
        }

        //cancel requests
        var newReq = await _context.Requests
            .Where(x => x.Status == LessonRequestStatus.New &&
                        (x.CreatedId == IdentityId || x.Tutor.Id == IdentityId))
            .ToListAsync();
        foreach (var request in newReq)
            request.Status = LessonRequestStatus.Rejected;

        //remove solutions
        var solutions = await _context.Solutions
            .Where(x => x.StudentId == IdentityId)
            .ToListAsync();
        _context.Solutions.RemoveRange(solutions);

        //save changes
        await _context.SaveChangesAsync();

        //logout
        await _signInManager.SignOutAsync();
        return Redirect("/");
    }
}
```

Файл SubjectController.cs

```
using Infra.DatabaseAdapter;
using Infra.DatabaseAdapter.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace Web.Controllers.Admin;

[Authorize(Roles = "Administrator")]
[Route("/[controller]/[action]")]
```

```
[Produces("text/html; charset=utf-8")]
public class SubjectController : Controller
{
    private readonly AppDbContext _context;

    public SubjectController(AppDbContext context) => _context = context;

    // GET: Subject
    [HttpGet]
    public async Task<IActionResult> Index() => View(await _context.Subjects.ToListAsync());

    // GET: Subject/Create
    [HttpGet]
    public IActionResult Create() => View();

    // POST: Subject/Create

    [HttpPost]
    public async Task<IActionResult> Create([Bind("Id,Name")] SubjectModel subjectModel)
    {
        if (ModelState.IsValid)
        {
            _context.Add(subjectModel);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        return View(subjectModel);
    }

    // GET: Subject/Edit/5
    [HttpGet]
    [Route("{id?}")]
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null) return NotFound();

        var subjectModel = await _context.Subjects.FindAsync(id);
        if (subjectModel == null) return NotFound();
        return View(subjectModel);
    }

    // POST: Subject/Edit/5

    [HttpPost]
    [Route("{id}")]
    public async Task<IActionResult> Edit(int id, [Bind("Id,Name")] SubjectModel subjectModel)
    {
        if (id != subjectModel.Id) return NotFound();

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(subjectModel);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!SubjectModelExists(subjectModel.Id))
                    return NotFound();
                else
                    throw;
            }

            return RedirectToAction(nameof(Index));
        }
```

```
        return View(subjectModel);
    }

    // GET: Subject/Delete/5
    [HttpGet]
    [Route("{id?}")]
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null) return NotFound();

        var subjectModel = await _context.Subjects
            .FirstOrDefaultAsync(m => m.Id == id);
        if (subjectModel == null) return NotFound();

        return View(subjectModel);
    }

    // POST: Subject/Delete/5
    [HttpPost]
    [ActionName("Delete")]
    [Route("{id}")]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        var subjectModel = await _context.Subjects.FindAsync(id);
        if (subjectModel != null) _context.Subjects.Remove(subjectModel);

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool SubjectModelExists(int id) => _context.Subjects.Any(e => e.Id == id);
}
```

Файл AssignmentController.cs

```
using System.Security.Claims;
using Domain.Commands;
using Domain.Helpers;
using Domain.Queries;
using Microsoft.AspNetCore.Mvc;
using MediatR;
using Microsoft.AspNetCore.Authorization;
using Web.Helpers;
using Web.Models.Assignments;

namespace Web.Controllers;

[Authorize]
[Route("/[controller]/[action]")]
[Produces("text/html; charset=utf-8")]
public class AssignmentController : Controller
{
    private readonly IMediator _mediator;
    private readonly ControllerHelpers _helper;
    public int IdentityId => Convert.ToInt32(User.FindFirstValue(ClaimTypes.NameIdentifier));

    public AssignmentController(IMediator mediator)
    {
        _mediator = mediator;
        _helper = new ControllerHelpers(mediator);
    }

    [ApiExplorerSettings(IgnoreApi = true)]
    public void AssignmentValidation(AssignmentVm model)
    {
        if (DateTime.Now > model.Assignment.Deadline)
```

```
            ModelState.AddModelError("Assignment.Deadline", "Не можна встановити термін здачі у минулому");
        if (model.Assignment.StudentsIds.Count == 0)
            ModelState.AddModelError("Assignment.StudentsIds", "Треба вибирати хоча б одного учня");
        if (model.Assignment.TutorId != IdentityId)
            ModelState.AddModelError("Assignment.TutorId", "Обрано невірного вчителя");
    }

    [HttpGet]
    public async Task<IActionResult> Index(GetAssignmentsQuery? filter)
    {
        var model = new AssignmentListVm();
        if (filter == null)
            filter = new GetAssignmentsQuery();
        filter.UserId = IdentityId;

        model.Assignments = await _mediator.Send(filter);
        model.Subjects = await _helper.GetSelectList(new GetAllSubjectsQuery());
        model.HisStudents = await _helper.GetSelectList(new GetTutorStudentsQuery() { TutorId = IdentityId });
        _helper.UpdateSelf(model.HisStudents, IdentityId);
        return View(model);
    }


    [HttpGet]
    public async Task<IActionResult> Create()
    {
        var model = new AssignmentVm();

        if (IdentityId == 0)
            return NotFound();
        //Отримання даних та наповнення моделі
        model.UserId = model.Assignment.TutorId = IdentityId;
        model.Subjects =
            await _helper.GetSelectList(new GetAllSubjectsQuery() { TutorId = IdentityId }, "Оберіть тематику");
        model.HisStudents = await _helper.GetSelectList(new GetTutorStudentsQuery() { TutorId = IdentityId });
        return View(model);
    }


    [HttpPost]
    public async Task<IActionResult> Create(AssignmentVm model)
    {
        AssignmentValidation(model);
        if (ModelState.IsValid)
        {
            if (model.Assignment.Description == null) // fix empty description
                model.Assignment.Description = "";

            var assignmentId = await _mediator.Send(new CreateAssignmentCommand()
            {
                CreatedId = IdentityId,
                Assignment = model.Assignment
            });
            return RedirectToAction(nameof(Edit), new { id = assignmentId });
        }

        model.UserId = model.Assignment.TutorId = IdentityId;
        model.Subjects =
            await _helper.GetSelectList(new GetAllSubjectsQuery() { TutorId = IdentityId }, "Оберіть тематику");
        model.HisStudents = await _helper.GetSelectList(new GetTutorStudentsQuery() { TutorId = IdentityId });
        return View(model);
    }

    [HttpGet]
    [Route("{id}")]
    public async Task<IActionResult> Edit(int id)
    {
```

```
        var model = new AssignmentVm();

        if (IdentityId == 0)
            return NotFound();

        var curAssignment = await _mediator.Send(new GetOneAssignmentQuery { AssignmentId = id, UserId = IdentityId
});

        if (curAssignment == null)
            return NotFound();

        model.UserId = IdentityId;
        model.Assignment = curAssignment;
        model.Subjects =
            await _helper.GetSelectList(new GetAllSubjectsQuery() { TutorId = IdentityId }, "Оберіть тематику");
        model.HisStudents = await _helper.GetSelectList(new GetTutorStudentsQuery() { TutorId = IdentityId });
        return View(model);
    }

    [HttpPost]
    [Route("{id}")]
    public async Task<IActionResult> Edit(int id, AssignmentVm model)
    {
        AssignmentValidation(model);
        if (id != model.Assignment.Id) return NotFound();

        if (ModelState.IsValid)
        {
            await _mediator.Send(new UpdateAssignmentCommand()
            {
                UpdatedBy = IdentityId,
                AssignmentId = model.Assignment.Id,
                Title = model.Assignment.Title,
                Description = model.Assignment.Description,
                Deadline = model.Assignment.Deadline,
                SubjectId = model.Assignment.SubjectId,
                StudentIds = model.Assignment.StudentsIds
            });
            return RedirectToAction(nameof(Index));
        }

        model.Subjects = await _helper.GetSelectList(new GetAllSubjectsQuery(), "Оберіть тематику");
        model.HisStudents = await _helper.GetSelectList(new GetTutorStudentsQuery() { TutorId = IdentityId });
        return View(model);
    }

    [HttpPost]
    [Route("{id}")]
    public async Task<ActionResult> Delete(int id)
    {
        Response.StatusCode = 400;
        if (id != 0)
        {
            var success = await _mediator.Send(new DeleteAssignmentCommand()
            {
                AssignmentId = id,
                TutorId = IdentityId
            });
            if (success)
                return RedirectToAction(nameof(Index));
        }

        return NotFound();
    }
}
```

Файл FilesController.cs

```
using System.Diagnostics;
using System.Security.Claims;
using Domain.Commands;
using Domain.Models;
using Domain.Queries;
using Humanizer;
using Infra;
using MediatR;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.StaticFiles;
using Web.Helpers;
using Web.Models.Files;
using Web.Models.Shared;

namespace Web.Controllers;

[Authorize]
[Route("/[controller]/[action]")]
public class FileController : Controller
{
    private readonly IMediator _mediator;
    public int IdentityId => Convert.ToInt32(User.FindFirstValue(ClaimTypes.NameIdentifier));

    public FileController(IMediator mediator) => _mediator = mediator;

    [HttpGet]
    public async Task<IActionResult> Download(string id)
    {
        if (id.Length != 36) //Довжина ключа GUID
            return Content("filename is not available");
        try
        {
            var file = await _mediator.Send(new DownloadFileQuery() { UserId = IdentityId, FileId = id });
            string? contentType;
            new FileExtensionContentTypeProvider().TryGetContentType(file.Filename, out contentType);
            return File(file.Memory, contentType ?? "", file.Filename);
        }
        catch
        {
            return NotFound();
        }
    }


    [HttpPost]
    public async Task<IActionResult> Upload(UploadFileVm model)
    {
        var helper = new ControllerHelpers(_mediator);
        try
        {
            if (model.UserId == null)
                model.UserId = IdentityId;
            else if (model.UserId != IdentityId)
                throw new Exception("Користувач вказаний невірно");
            var command = new UploadFileCommand()
            {
                UserId = model.UserId,
                AssignmentId = model.AssignmentId,
                SolutionId = model.SolutionId,
                FileBytes = await helper.BytesFromFormFile(model.FormFile),
                FileName = model.FormFile.FileName
            };
            await _mediator.Send(command);
            return Ok();
        }
        catch (Exception e)
```

```
        {
            return BadRequest(e.Message);
        }
    }
}
```

Файл HomeController.cs

```
using System.Diagnostics;
using MediatR;
using Microsoft.AspNetCore.Mvc;
using Web.Models;
using Web.Models.Shared;

namespace Web.Controllers;

public class HomeController : Controller
{
    public IActionResult Privacy()
    {
        var result = View();
        return result;
    }

    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
    public IActionResult Error(string msg = "") => View(new ErrorVm
    {
        RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier,
        Message = msg
    });

    public IActionResult About() => View();

    public IActionResult InDevelopment() => View();
}
```

Файл LessonController.cs

```
using System.Security.Claims;
using Domain.Commands;
using Domain.Helpers;
using Domain.Queries;
using Humanizer;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using MediatR;
using Microsoft.AspNetCore.Authorization;
using Web.Helpers;
using Web.Models.Lessons;

namespace Web.Controllers;

[Authorize]
[Route("/[controller]/[action]")]
public class LessonController : Controller
{
    private readonly IMediator _mediator;
    private readonly ControllerHelpers _helper;
    public int IdentityId => Convert.ToInt32(User.FindFirstValue(ClaimTypes.NameIdentifier));

    public LessonController(IMediator mediator)
    {
        _mediator = mediator;
        _helper = new ControllerHelpers(mediator);
    }

    [ApiExplorerSettings(IgnoreApi = true)]
```

```
    private void LessonValidation(LessonVm model)
    {
        if (model.Lesson.TutorId != IdentityId)
            ModelState.AddModelError("Lesson.TutorId", "Обрано невірного вчителя");
        if (model.Lesson.From.Date != model.Lesson.To.Date)
            ModelState.AddModelError("Lesson.To", "Зустріч має проходити протягом дня");
        if (model.Lesson.From.Date > model.Lesson.To.Date)
            ModelState.AddModelError("Lesson.To", "Можливо ви переплутали час місцями");
        if (model.Lesson.From < DateTime.Now.AddHours(-1))
            ModelState.AddModelError("Lesson.To", " Ви не можете створити подію в минулому");
        if (model.Lesson.StudentsIds.Count == 0)
            ModelState.AddModelError("Lesson.StudentsIds", "Треба вибирати хоча б одного учня");
    }


    [HttpGet]
    public async Task<IActionResult> Index()
    {
        var lessons = new ListLessonVm();
        lessons.Lessons = await _mediator.Send(new GetLessonsQuery()
        {
            StudentId = IdentityId,
            TutorId = IdentityId
        });
        lessons.IsTutor = await _mediator.Send(new UserIsTutorQuery() { UserId = IdentityId });
        lessons.UserId = IdentityId;
        return View(lessons);
    }

    [HttpGet]
    public async Task<IActionResult> Create()
    {
        var model = new LessonVm();

        if (IdentityId == 0)
            return NotFound();

        var userData = await _mediator.Send(new GetUserProfileQuery() { ProfileId = IdentityId });
        model.Lesson.TutorName = userData.FullName;
        model.Lesson.From = DateTime.Now.Date.AddHours(DateTime.Now.Hour).AddMinutes(DateTime.Now.Minute);
        model.Lesson.To = model.Lesson.From.AddHours(1);
        model.Lesson.TutorId = model.UserId = IdentityId;
        model.Subjects =
            await _helper.GetSelectList(new GetAllSubjectsQuery() { TutorId = IdentityId }, "Оберіть тематику");
        model.HisStudents = await _helper.GetSelectList(new GetTutorStudentsQuery() { TutorId = model.Lesson.TutorId
});
        return View(model);
    }


    [HttpPost]
    public async Task<IActionResult> Create(LessonVm model)
    {
        LessonValidation(model);

        if (ModelState.IsValid)
        {
            if (model.Lesson.Comment == null)
                model.Lesson.Comment = "";
            await _mediator.Send(new CreateLessonCommand() { CreatedId = IdentityId, Lesson = model.Lesson });
            return RedirectToAction(nameof(Index));
        }

        model.Lesson.TutorId = model.UserId = IdentityId;
        model.Subjects =
            await _helper.GetSelectList(new GetAllSubjectsQuery() { TutorId = IdentityId }, "Оберіть тематику");
```

```
        model.HisStudents = await _helper.GetSelectList(new GetTutorStudentsQuery() { TutorId = model.Lesson.TutorId
});
        return View(model);
    }


    [HttpGet]
    public async Task<IActionResult> Details(int id)
    {
        var model = new LessonVm();

        if (IdentityId == 0)
            return NotFound();

        var curLesson = await _mediator.Send(new GetOneLessonQuery { LessonId = id, UserId = IdentityId });

        if (curLesson == null)
            return NotFound();

        model.UserId = IdentityId;
        model.Lesson = curLesson;
        if (model.Lesson.TutorId == IdentityId)
            model.IsTutor = true;
        model.Subjects = await _helper.GetSelectList(new GetAllSubjectsQuery(), "Оберіть тематику");
        model.HisStudents = await _helper.GetSelectList(new GetTutorStudentsQuery() { TutorId = model.Lesson.TutorId
});
        return View(model);
    }

    [HttpGet]
    public async Task<IActionResult> Delete(int? id)
    {
        if (!id.HasValue)
            return NotFound();

        var tempLesson = await _mediator.Send(new GetOneLessonQuery { LessonId = id.Value, UserId = IdentityId });
        if (tempLesson == null)
            return NotFound();
        else
            return View(new LessonVm() { Lesson = tempLesson });
    }

    [HttpPost]
    public async Task<ActionResult> Delete(int id)
    {
        Response.StatusCode = 400;
        if (id != 0)
        {
            var success =
                await _mediator.Send(new DeleteLessonCommand() { LessonId = id, TutorId = IdentityId });
            if (success)
                return RedirectToAction(nameof(Index));
        }

        return RedirectToAction(nameof(Index));
    }
}
```

Файл LessonRequestController.cs

```
using System.Runtime.InteropServices.JavaScript;
using System.Security.Claims;
using MediatR;
using Microsoft.AspNetCore.Mvc;
using Web.Controllers;
using Domain.Queries;
using Domain.Commands;
```

```
using Domain.Helpers;
using Domain.Models;
using Infra.DatabaseAdapter.Helpers;
using Microsoft.AspNetCore.Authorization;
using Web.Helpers;
using Web.Models.LessonRequest;

namespace Web.Controllers;

[Authorize]
[Route("/[controller]/[action]")]
public class LessonRequestController : Controller
{
    private readonly IMediator _mediator;
    private readonly ControllerHelpers _helper;
    public int IdentityId => Convert.ToInt32(User.FindFirstValue(ClaimTypes.NameIdentifier));

    public LessonRequestController(IMediator mediator)
    {
        _mediator = mediator;
        _helper = new ControllerHelpers(mediator);
    }

    [HttpGet]
    public async Task<IActionResult> Index(bool onlyActive = false)
    {
        var vm = new LessonRequestVm
        {
            MyRequests =
                await _mediator.Send(new GetUserRequestsQuery { UserId = IdentityId, IsTutor = false }),
            RequestsForMe = await _mediator.Send(new GetUserRequestsQuery
                { UserId = IdentityId, IsTutor = true })
        };
        return View(vm);
    }

    [HttpPost]
    public async Task<IActionResult> Create(CreateRequestCommand command)
    {
        try
        {
            if (!command.From.HasValue || !command.To.HasValue)
            {
                ModelState.AddModelError("From", "Треба обрати час");
            }
            else
            {
                if (command.From.Value < DateTime.Now.AddHours(-1))
                    ModelState.AddModelError("From", " Ви не можете створити подію в минулому");
                if (command.From.Value.Date != command.To.Value.Date)
                    ModelState.AddModelError("Lesson.To", "Зустріч має проходити протягом дня");
                if (command.From.Value.Date > command.To.Value.Date)
                    ModelState.AddModelError("Lesson.To", "Можливо ви переплутали час місцями");
            }

            if (command.CreatedId == 0)
                command.CreatedId = IdentityId;
            else if (command.CreatedId != IdentityId)
                throw new Exception($"command.CreatedBy={command.CreatedId},app.UserId={IdentityId}");

            if (ModelState.IsValid)
            {
                var result = await _mediator.Send(command);

                return Json(result);
            }
        }
```

```
        catch (Exception e)
        {
            ModelState.AddModelError("", e.Message);
        }

        var errList = (from item in ModelState
            where item.Value.Errors.Any()
            select item.Value.Errors[0].ErrorMessage).ToList();
        return StatusCode(500, errList);
    }

    [HttpGet]
    public async Task<IActionResult> Details(int id)
    {
        if (IdentityId == 0)
            return NotFound();

        var requestsForMe = await _mediator.Send(new GetUserRequestsQuery { UserId = IdentityId, IsTutor = true });
        var model = requestsForMe.Where(x => x.Id == id).FirstOrDefault();
        if (model == null)
            return RedirectToAction(nameof(Index));
        else
            return View(model);
    }


    [HttpPost]
    public async Task<IActionResult> Update(LessonRequest model)
    {
        var command = new UpdateRequestCommand()
        {
            Id = model.Id,
            UpdatedBy = IdentityId,
            TutorId = model.TutorId,
            Status = (LessonRequestStatus)int.Parse(Request.Form["Status"][1]),
            TutorComment = model.TutorComment,
            From = model.From,
            To = model.To
        };
        await _mediator.Send(command);
        return RedirectToAction(nameof(Index));
    }
}
```

Файл ProfileController.cs

```
using System.Security.Claims;
using AutoMapper;
using AutoMapper.Configuration.Annotations;
using Domain.Commands;
using Domain.Queries;
using Domain.Helpers;
using Infra.DatabaseAdapter.Models;
using MediatR;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.ModelBinding;
using Microsoft.AspNetCore.Mvc.Rendering;
using Web.Helpers;
using Web.Models.Shared;
using Web.Models.Profile;

namespace Web.Controllers;

[Authorize]
[Route("/[controller]/[action]")]
public class ProfileController : Controller
```

```
{
    private readonly IMediator _mediator;
    private readonly ControllerHelpers _helper;
    public int IdentityId => Convert.ToInt32(User.FindFirstValue(ClaimTypes.NameIdentifier));

    public ProfileController(IMediator mediator)
    {
        _mediator = mediator;
        _helper = new ControllerHelpers(mediator);
    }


    [ApiExplorerSettings(IgnoreApi = true)]
    [AllowAnonymous]
    [HttpGet]
    [Route("/")]
    [Route("/[controller]/[action]")]
    public async Task<IActionResult> Index(SearchVm model)
    {
        //Main Lists
        model.Subjects = await _helper.GetSelectList(new GetAllSubjectsQuery(), "Оберіть тематику");
        model.Cities = await _helper.GetSelectList(new GetAllCitiesQuery(), "Оберіть населений пункт");

        //Add Cards
        model.Filters.IdentityId = IdentityId;
        var userIds = await _mediator.Send(model.Filters);
        foreach (var userId in userIds)
        {
            var card = new TutorCardVm()
            {
                Subjects = model.Subjects,
                TutorVm = await _mediator.Send(new GetTutorProfileQuery { ProfileId = userId }),
                UserVm = await _mediator.Send(new GetUserProfileQuery { ProfileId = userId })
            };
            card.UserVm.CityId =
                model.Cities.FirstOrDefault(x => x.Value == card.UserVm.CityId)?.Text ?? "Не вказано";
            model.TutorCards.Add(card);
        }

        //Result
        return View(model);
    }

    [HttpGet]
    [AllowAnonymous]
    [Route("{id}")]
    public async Task<IActionResult> Details(int id = 0)
    {
        if (id == 0)
            id = IdentityId;

        var model = await GetUserModel(id, true);
        model.Reviews = await _mediator.Send(new GetTutorReviewsQuery() { TutorId = id });
        model.CreateRequestCommand.CreatedId = IdentityId;
        return View(model);
    }

    [ApiExplorerSettings(IgnoreApi = true)]
    [HttpGet]
    public async Task<ProfileVm> GetUserModel(int userId, bool onlyTutorData = false)
    {
        ProfileVm model = new() { IdentityId = IdentityId };
        model.Cities = await _helper.GetSelectList(new GetAllCitiesQuery(), "Оберіть населений пункт");
        model.UserVm = await _mediator.Send(new GetUserProfileQuery { ProfileId = userId });
        if (model.UserVm.ProfileEnabled)
        {
            var q = new GetAllSubjectsQuery();
```

```
        model.Subjects = await _helper.GetSelectList(q, "Оберіть тематику");
        model.TutorVm = await _mediator.Send(new GetTutorProfileQuery { ProfileId = userId });
        model.CreateRequestCommand.TutorId = userId;

        //Необхідний для показу списку предметів
        if (onlyTutorData)
            q.TutorId = userId;
        model.CreateRequestCommand.Subjects = await _helper.GetSelectList(q);
    }

    return model;
}

[HttpGet]
public async Task<IActionResult> Edit()
{
    var model = await GetUserModel(IdentityId);
    return View(model);
}

[HttpPost]
public async Task<IActionResult> Edit(ProfileVm model)
{
    // Ігнорувати помилки форми якщо не вчитель
    if (!model.UserVm.ProfileEnabled)
        foreach (var state in ModelState)
            if (state.Key.StartsWith(nameof(model.TutorVm)))
                state.Value.ValidationState = ModelValidationState.Skipped;

    if (ModelState.IsValid)
    {
        await _mediator.Send(new UpdateUserCommand { Profile = model.UserVm });

        //Upload new avatar
        if (model.NewAvatar != null)
        {
            var imgBytes = await _helper.BytesFromFormFile(model.NewAvatar);
            await _mediator.Send(new SaveAvatarCommand() { UserId = IdentityId, ImageBytes = imgBytes });
        }

        if (model.UserVm.ProfileEnabled)
        {
            if (model.TutorVm == null)
                model.TutorVm = await _mediator.Send(new GetTutorProfileQuery { ProfileId = model.UserVm.Id });
            if (model.TutorVm.Id == 0)
                model.TutorVm.Id = model.UserVm.Id;
            await _mediator.Send(new UpdateTutorCommand { Profile = model.TutorVm });
            model.Subjects = await _helper.GetSelectList(new GetAllSubjectsQuery(), "Оберіть предмет");
        }
    }

    model.Cities = await _helper.GetSelectList(new GetAllCitiesQuery(), "Оберіть населений пункт");
    return View(model);
}

[HttpGet]
public async Task<IActionResult> IsFavorite(CheckFavoriteQuery query)
{
    var isChecked = await _mediator.Send(query);
    return Json(new { status = isChecked });
}

[HttpPost]
public async Task<IActionResult> IsFavorite(CheckFavoriteCommand command)
{
    try
    {
```

```
            if (command.UserId != IdentityId)
                throw new Exception($"command.UserId={command.UserId},app.UserId={IdentityId}");
            var isChecked = await _mediator.Send(command);
            return Json(new { status = isChecked });
        }
        catch (Exception e)
        {
            return BadRequest(e.Message);
        }
    }
}
```

Файл ReviewController.cs

```
using System.Security.Claims;
using Domain.Commands;
using Domain.Helpers;
using Domain.Models;
using Domain.Queries;
using Microsoft.AspNetCore.Mvc;
using MediatR;
using Microsoft.AspNetCore.Authorization;
using Web.Helpers;
using Web.Models.Assignments;
using Web.Models.Profile;
using ZstdSharp.Unsafe;

namespace Web.Controllers;

[Authorize]
[Route("/[controller]/[action]")]
public class ReviewController : Controller
{
    private readonly IMediator _mediator;
    private readonly ControllerHelpers _helper;
    public int IdentityId => Convert.ToInt32(User.FindFirstValue(ClaimTypes.NameIdentifier));

    public ReviewController(IMediator mediator)
    {
        _mediator = mediator;
        _helper = new ControllerHelpers(mediator);
    }


    [HttpGet]
    [Route("{tutorId}")]
    public async Task<IActionResult> Edit(int tutorId)
    {
        if (IdentityId == 0)
            return NotFound();
        var userReview = await _mediator.Send(new GetReviewQuery { TutorId = tutorId, AuthorId = IdentityId });
        return View(userReview);
    }


    [HttpPost]
    [Route("{tutorId}")]
    public async Task<IActionResult> Edit(int tutorId, Review model)
    {
        model.TutorId = tutorId;
        if (model.AuthorId != IdentityId)
            ModelState.AddModelError("AuthorId", "Автор вказаний невірно");
        if (model.Rating == 0) ModelState.AddModelError("Rating", "Оцінка обов'язкова");

        if (ModelState.IsValid)
            try
            {
```

```
            model = await _mediator.Send(new AddOrUpdateReviewCommand() { Review = model });
            return RedirectToRoute(new
            {
                controller = "Profile",
                action = "Details",
                id = model.TutorId
            });
        }
        catch (Exception e)
        {
            ModelState.AddModelError("AuthorId", e.Message);
        }

    return View(model);
  }
}
```

Файл SessionController.cs

```
using System.Security.Claims;
using AutoMapper;
using Domain.Commands;
using Domain.Queries;
using Domain.Helpers;
using MediatR;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Web.Models.Lessons;
using Web.Models.Shared;

namespace Web.Controllers;

[Authorize]
[Route("/[controller]/[action]")]
[Produces("application/json")]
public class SessionController : Controller
{
  private readonly IMediator _mediator;
  public int IdentityId => Convert.ToInt32(User.FindFirstValue(ClaimTypes.NameIdentifier));

  public SessionController(IMediator mediator) => _mediator = mediator;


  [HttpPost]
  public async Task<ActionResult> Create(CreateSessionCommand command)
  {
    try
    {
      if (command.CreatedBy == 0)
        command.CreatedBy = IdentityId;
      else if (command.CreatedBy != IdentityId)
        throw new Exception($"command.CreatedBy={command.CreatedBy},app.UserId={IdentityId}");
      var id = await _mediator.Send(command);
      return Json(id);
    }
    catch (Exception e)
    {
      return StatusCode(500, e);
    }
  }

  [HttpPost]
  public async Task<ActionResult> Delete(int id)
  {
    try
    {
      await _mediator.Send(new DeleteSessionCommand() { UpdatedBy = IdentityId, EventId = id });
```

```
            return Json(1);
        }
        catch
        {
            return StatusCode(500);
        }
    }


    [HttpGet]
    [AllowAnonymous]
    public async Task<IActionResult> List(GetEventQuery filter)
    {
        try
        {
            var times = await _mediator.Send(filter);
            return Json(times);
        }
        catch
        {
            return StatusCode(500);
        }
    }
}
```

Файл SolutionController.cs

```
using System.Security.Claims;
using Domain.Commands;
using Domain.Helpers;
using Domain.Queries;
using Infra.DatabaseAdapter.Helpers;
using Microsoft.AspNetCore.Mvc;
using MediatR;
using Microsoft.AspNetCore.Authorization;
using Web.Helpers;
using Web.Models.Assignments;
using Web.Models.Solutions;

namespace Web.Controllers;

[Authorize]
[Route("/[controller]/[action]")]
public class SolutionController : Controller
{
    private readonly IMediator _mediator;
    private readonly ControllerHelpers _helper;
    public int IdentityId => Convert.ToInt32(User.FindFirstValue(ClaimTypes.NameIdentifier));

    public SolutionController(IMediator mediator)
    {
        _mediator = mediator;
        _helper = new ControllerHelpers(mediator);
    }

    [HttpGet]
    public async Task<IActionResult> Index(GetSolutionsQuery? filter)
    {
        var model = new SolutionListVm();


        if (filter == null) filter = new GetSolutionsQuery();

        filter.UserId = IdentityId;
        model.Solutions = await _mediator.Send(filter);
        model.Subjects = await _helper.GetSelectList(new GetAllSubjectsQuery());
        model.HisStudents = await _helper.GetSelectList(new GetTutorStudentsQuery() { TutorId = IdentityId });
```

```
        _helper.UpdateSelf(model.HisStudents, IdentityId);


        model.HisTutors = await _helper.GetSelectList(new GetStudentTutorsQuery() { StudentId = IdentityId });
        _helper.UpdateSelf(model.HisTutors, IdentityId);
        model.Assignments = await _helper.GetSelectList(new GetAssignmentNamesQuery()
        {
            TutorId = IdentityId,
            StudentId = IdentityId
        });
        return View(model);
    }


    [HttpGet]
    [Route("{id}")]
    public async Task<IActionResult> Edit(int id)
    {
        var model = new SolutionVm();

        if (IdentityId == 0) return NotFound();

        var curSolution = await _mediator.Send(new GetOneSolutionQuery { SolutionId = id, UserId = IdentityId });

        if (curSolution == null) return NotFound();
        model.UserId = IdentityId;
        model.Solution = curSolution;
        model.IsTutor = IdentityId == model.Solution.TutorId;
        model.Subjects = await _helper.GetSelectList(new GetAllSubjectsQuery(), "Оберіть тематику");
        return View(model);
    }

    [HttpPost]
    [Route("{id}")]
    public async Task<IActionResult> Edit(int id, SolutionVm model)
    {
        if (model.Solution.SolutionFiles.Count > 5)
            ModelState.AddModelError("Solution.SolutionFiles", "До 5 файлів доступно для завантаження");

        if (id != model.Solution.Id) return NotFound();

        if (ModelState.IsValid)
        {
            //Оновити статус рішення якщо учень відправив відповідь
            if (!model.IsTutor && model.Solution.Answer != null && model.Solution.Answer.Length > 0)
                model.Solution.Status = SolutionStatus.Review;

            await _mediator.Send(new UpdateSolutionCommand()
            {
                UpdatedBy = IdentityId,
                SolutionId = model.Solution.Id,
                Status = model.Solution.Status,
                Answer = model.Solution.Answer,
                TutorComment = model.Solution.TutorComment
            });
            return RedirectToAction(nameof(Index));
        }

        model.IsTutor = IdentityId == model.Solution.TutorId;
        model.Subjects = await _helper.GetSelectList(new GetAllSubjectsQuery(), "Оберіть тематику");
        return View(model);
    }
}
```

Файл ControllerHelpers.cs

```
using MediatR;
```

```
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Helpers;

public class ControllerHelpers
{
    private readonly IMediator _mediator;

    public ControllerHelpers(IMediator mediator) =>
        _mediator = mediator;

    public async Task<List<SelectListItem>> GetSelectList(IRequest<Dictionary<int, string>> q, string defaultText = "")
    {
        var query = await _mediator.Send(q);
        var list = query
            .Select(o => new SelectListItem { Value = o.Key.ToString(), Text = o.Value })
            .ToList();
        if (defaultText != "")
            list.Insert(0, new SelectListItem(defaultText, "0"));
        return list;
    }

    public void UpdateSelf(List<SelectListItem> inputs, int userId)
    {
        var sserIdStr = userId.ToString();
        var item = inputs.FirstOrDefault(x => x.Value == sserIdStr);
        if (item == null)
            inputs.Insert(0, new SelectListItem { Value = sserIdStr, Text = "Мої" });
        else
            item.Text = "Мої";
    }

    public async Task<byte[]> BytesFromFormFile(IFormFile file)
    {
        using var ms = new MemoryStream();
        await file.CopyToAsync(ms);
        return ms.ToArray();
    }
}
```

Файл StringExtensions.cs

```
namespace Web.Helpers;

public static class StringExtensions
{
    public static string Truncate(this string value, int maxLength) =>
        value.Length <= maxLength ? value : value.Substring(0, maxLength) + "...";
}
```

Файл RemovedRoleMiddleware.cs

```
namespace Web.Middlewares;

public class RemovedRoleMiddleware
{
    private readonly RequestDelegate _next;
    private const string Path = "/RemoveUser/Removed";

    public RemovedRoleMiddleware(RequestDelegate next) => _next = next;

    public async Task Invoke(HttpContext context)
    {
        // Перевірка автентифікований, ролі та сторінки
        if (context.User.Identity.IsAuthenticated && context.User.IsInRole("Removed"))
            if (!context.Request.Path.HasValue || !context.Request.Path.Value.StartsWith("/RemoveUser"))
            {
```

```
            context.Response.Redirect(Path);
            return;
        }


    await _next(context);
  }
}
```

Файл AssignmentListVm.cs

```
using Domain.Models;
using Domain.Queries;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Models.Assignments;

public class AssignmentListVm
{
  public GetAssignmentsQuery Filter { get; set; } = new();
  public List<Assignment> Assignments { get; set; } = [];

  public List<SelectListItem> Subjects { get; set; } = [];
  public List<SelectListItem> HisStudents { get; set; } = [];
}
```

Файл AssignmentVm.cs

```
using System.ComponentModel;
using Domain.Models;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Models.Assignments;

public class AssignmentVm
{
  public int? UserId { get; set; }
  public Assignment Assignment { get; set; } = new();
  public List<SelectListItem> Subjects { get; set; } = [];

  public List<SelectListItem> HisStudents { get; set; } = [];
}
```

Файл UploadFileVm.cs

```
namespace Web.Models.Files;

public class UploadFileVm
{
  public int? UserId { get; set; }
  public IFormFile FormFile { get; set; } = null!;
  public int? AssignmentId { get; set; }
  public int? SolutionId { get; set; }
}
```

Файл LessonRequestVm.cs

```
using Domain.Commands;
using Domain.Models;
using Infra.DatabaseAdapter.Helpers;

namespace Web.Models.LessonRequest;

public class LessonRequestVm
{
  public List<Domain.Models.LessonRequest> MyRequests { get; set; } = [];
  public List<Domain.Models.LessonRequest> RequestsForMe { get; set; } = [];
```

```
    public UpdateRequestCommand RequestUpdate { get; set; } = new();
}
```

Файл LessonVm.cs

```
using Domain.Models;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Models.Lessons;

public class LessonVm
{
    public int UserId { get; set; }
    public bool IsTutor { get; set; }
    public List<SelectListItem> Subjects { get; set; } = [];
    public Lesson Lesson { get; set; } = new();
    public List<SelectListItem> HisStudents { get; set; } = [];
}
```

Файл ListLessonVm.cs

```
using Domain.Models;

namespace Web.Models.Lessons;

public class ListLessonVm
{
    public int UserId { get; set; }
    public bool IsTutor { get; set; }
    public List<Lesson> Lessons { get; set; } = [];
}
```

Файл ProfileVm.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using Domain.Commands;
using Domain.Models;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Models.Profile;

public class ProfileVm : TutorCardVm
{
    [Required] public int IdentityId { get; set; }
    public List<SelectListItem> Cities { get; set; } = [];

    public CreateRequestCommand CreateRequestCommand { get; set; } = new();

    [DisplayName("Завантажити новий аватар")]
    public IFormFile? NewAvatar { get; set; }

    public List<Review> Reviews { get; set; } = [];
}
```

Файл SearchVm.cs

```
using Domain.Queries;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Models.Profile;

public class SearchVm
{
    public List<SelectListItem> Subjects { get; set; } = [];
    public List<SelectListItem> Cities { get; set; } = [];
    public List<TutorCardVm> TutorCards { get; set; } = [];
    public GetTutorsQuery Filters { get; set; } = new();
```

```
}
```

Файл TutorCardVm.cs

```
using Domain.Models;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Models.Profile;

public class TutorCardVm
{
    public User UserVm { get; set; } = null!;
    public Tutor? TutorVm { get; set; } = null!;
    public List<SelectListItem> Subjects { get; set; } = [];
}
```

Файл RemoveUserVm.cs

```
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Models.RemoveUser;

public class RemoveUserVm
{
    [Display(Name = "Користувач")] public string UserId { get; set; } = null!;
    public List<SelectListItem> UserList { get; set; } = new();
}
```

Файл CheckboxVm.cs

```
namespace Web.Models.Shared;

public class CheckboxVm
{
    public int Id { get; set; }
    public string LabelName { get; set; } = "";
    public bool IsChecked { get; set; }
}
```

Файл ErrorVm.cs

```
namespace Web.Models.Shared;

public class ErrorVm
{
    public string? RequestId { get; set; }

    public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

    public bool Status { get; set; }
    public string Message { get; set; } = string.Empty;
}
```

Файл SolutionListVm.cs

```
using Domain.Models;
using Domain.Queries;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Models.Solutions;

public class SolutionListVm
{
    public GetSolutionsQuery Filter { get; set; } = new();
    public List<Solution> Solutions { get; set; } = [];

    public List<SelectListItem> Subjects { get; set; } = [];
```

```
    public List<SelectListItem> HisStudents { get; set; } = [];
    public List<SelectListItem> HisTutors { get; set; } = [];
    public List<SelectListItem> Assignments { get; set; } = [];
}
```

Файл SolutionVm.cs

```
using Domain.Models;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace Web.Models.Solutions;

public class SolutionVm
{
    public int? UserId { get; set; }
    public Solution Solution { get; set; } = new();
    public List<SelectListItem> Subjects { get; set; } = [];
    public bool IsTutor { get; set; }
}
```

Файли завдання

Файл Create.cshtml

```
@model Web.Models.Assignments.AssignmentVm

@{
    ViewData["Title"] = "Додати нове завдання";
    Layout = "_LayoutFormSm";
}


<form asp-action="Create">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <input asp-for="Assignment.TutorId" hidden/>
    <div class="form-group">
        <label asp-for="Assignment.Title" class="control-label"></label>
        <input asp-for="Assignment.Title" class="form-control"/>
        <span asp-validation-for="Assignment.Title" class="text-danger"></span>
    </div>
    <div class=" form-group">
        <label asp-for="Assignment.SubjectId" class="form-check-label mt-2"></label>
        <select asp-for="Assignment.SubjectId" class="form-select" asp-items="@Model.Subjects"></select>
        <span asp-validation-for="Assignment.SubjectId" class="text-danger"></span>
    </div>
    <div class=" form-group">
        <label asp-for="Assignment.Deadline" class="control-label"></label>
        <input asp-for="Assignment.Deadline" class="form-control"/>
        <span asp-validation-for="Assignment.Deadline" class="text-danger"></span>
    </div>
    <div class="form-group">
        <div class="input-group">
            <label asp-for="Assignment.StudentsIds" class="form-check-label mt-2"></label>
            <select asp-for="Assignment.StudentsIds" asp-items="@Model.HisStudents" class="selectpicker  form-control"
data-width="100%" data-live-search="true" multiple></select>
            <span asp-validation-for="Assignment.StudentsIds" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <label asp-for="Assignment.Description" class="control-label"></label>
        <textarea asp-for="Assignment.Description" rows="4" class="form-control"></textarea>
        <span asp-validation-for="Assignment.Description" class="text-danger"></span>
    </div>

    <div class="d-flex">
        <div class="ml-auto px-2">
```

```
            <input type="submit" value="Зберегти" class="btn btn-primary"/>
        </div>
        <div>
            <a class="btn btn-secondary " asp-action="Index">Скасувати</a>
        </div>
    </div>
</form>
```

Файл Edit.cshtml

```
@model Web.Models.Assignments.AssignmentVm

@{
    ViewData["Title"] = "Завдання #" + Model.Assignment.Id;
    Layout = "_Layout";
}

<div class="container-sm col">
    <div class="row h-100 d-flex align-items-center justify-content-center">
        <div class="col-lg-6">
            <div class="bg-white rounded shadow-sm p-5 mb-4">
                <h4 class="mb-3 text-center">@ViewData["Title"]</h4>
                <form asp-action="Edit">
                    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
                    <input type="hidden" asp-for="Assignment.Id"/>
                    <input asp-for="Assignment.TutorId" hidden/>
                    <div class="form-group">
                        <label asp-for="Assignment.Title" class="control-label"></label>
                        <input asp-for="Assignment.Title" class="form-control"/>
                        <span asp-validation-for="Assignment.Title" class="text-danger"></span>
                    </div>
                    <div class=" form-group">
                        <label asp-for="Assignment.SubjectId" class="form-check-label mt-2"></label>
                        <select asp-for="Assignment.SubjectId" class="form-select" asp-items="@Model.Subjects"></select>
                        <span asp-validation-for="Assignment.SubjectId" class="text-danger"></span>
                    </div>
                    <div class=" form-group">
                        <label asp-for="Assignment.Deadline" class="control-label"></label>
                        <input asp-for="Assignment.Deadline" class="form-control" type="datetime-local"/>
                        <span asp-validation-for="Assignment.Deadline" class="text-danger"></span>
                    </div>
                    <div class="form-group">
                        <div class="input-group">
                            <label asp-for="Assignment.StudentsIds" class="form-check-label mt-2"></label>
                            <select asp-for="Assignment.StudentsIds" asp-items="@Model.HisStudents" class="selectpicker
form-control" data-width="100%" data-live-search="true" multiple></select>
                            <span asp-validation-for="Assignment.StudentsIds" class="text-danger"></span>
                        </div>
                    </div>
                    <div class="form-group">
                        <label asp-for="Assignment.Description" class="control-label"></label>
                        <textarea asp-for="Assignment.Description" rows="4" class="form-control"></textarea>
                        <span asp-validation-for="Assignment.Description" class="text-danger"></span>
                    </div>
                    <div class="d-flex">
                        <div class="mr-auto">
                            <button asp-action="Delete" asp-route-id="@Model.Assignment.Id" class="btn btn-outline-danger">
                                <i class="fa-solid fa-trash-can"></i>
                            </button>
                        </div>
                        <div class="px-2">
                            <input type="submit" value="Зберегти" class="btn btn-primary"/>
                        </div>
                        <div>
                            <a class="btn btn-secondary " asp-action="Index">Скасувати</a>
                        </div>
```

```
            </div>
          </form>


        </div>
      </div>
    </div>
</div>

<!-- Assignment Files -->
<div class="container-sm">
    <div class="row h-100 d-flex align-items-center justify-content-center">
        <div class="col-lg-6">
            <div class="bg-white rounded shadow-sm p-5 mb-4">
                <partial name="FilesForm" model="@Model.Assignment.FileNames"/>
                <form id="uploadform" asp-controller="File" asp-action="Upload" enctype="multipart/form-data">
                    <input type="hidden" class="mt-3" asp-for="Assignment.Id" name="assignmentId"/>
                    <input class="form-control form-control-sm" id="uploadform-file" name="FormFile" type="file">
                </form>
            </div>
        </div>
    </div>
</div>
<!-- End Assignment Files -->
```

Файл Index.cshtml

```
@using Domain.Queries
@using Web.Helpers
@model Web.Models.Assignments.AssignmentListVm

@{
    ViewData["Title"] = "Мої завдання";
    Layout = "_LayoutFilterTable";
}

<!-- Head button -->

<div class="d-flex mb-3">
    <div class="p2">
        <h4 class="mb-1">@ViewData["Title"]</h4>
    </div>
    <div class="ml-auto">
        <a asp-action="Create" class=" text-whity">
            Додати завдання
            <i class="fas fa-plus btn btn-primary ml-1"></i>
        </a>
    </div>
</div>
<!-- End Head button -->

<!-- Head Filter-->
<form asp-action="Index" method="get" class="my-3">

    <div class="row d-flex row-cols-md-6 justify-content-start">
        <input asp-for="Filter.UserId" name="userid" hidden/>
        <div class="col">
            <label asp-for="Filter.SubjectId" class="form-check-label"></label>
            <select asp-for="Filter.SubjectId" asp-items="@Model.Subjects" class="selectpicker form-control" data-
width="100%" data-live-search="true" multiple></select>
        </div>
        <div class="col">
            <label asp-for="Filter.StudentId" class="form-check-label"></label>
            <select asp-for="Filter.StudentId" asp-items="@Model.HisStudents" class="selectpicker form-control" data-
width="100%" data-live-search="true" multiple></select>
        </div>
        <div class="col align-self-end">
```

```
            <button type="submit" class="btn btn-primary">
                <i class="fa-solid fa-arrows-rotate"></i>
            </button>
        </div>
    </div>
</form>
<!-- End Head Filter-->

<!-- Table-->
<table class=" table pretty-table">
    <thead>
    <tr>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Assignments.First().Id)
        </th>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Assignments.First().Title)
        </th>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Assignments.First().SubjectName)
        </th>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Assignments.First().Deadline)
        </th>
        <th scope="col"></th>
    </tr>
    </thead>
    <tbody>
    @foreach (var item in Model.Assignments)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Id)
            </td>
            <td>
                @item.Title.Truncate(25)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.SubjectName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Deadline)
            </td>
            <td>
                <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-outline-warning">
                    <i class="fa-solid fa-pen-to-square"></i>
                </a>
                @{
                    var filter = new Dictionary<string, string> { { "Filter.AssignmentId", item.Id.ToString() } };
                }
                <a asp-controller="Solution" asp-action="Index"
                  asp-all-route-data="filter" class="btn btn-outline-primary">
                    <i class="fa-solid fa-graduation-cap"></i>
                </a>
            </td>
        </tr>
    }
    </tbody>
</table>
<!-- End Table-->
```

Сторінки міста(сторінки предметів побудовані за аналогією)

Файл Create.cshtml

```
@model Infra.DatabaseAdapter.Models.CityModel
```

```
@{
    ViewData["Title"] = "Додавання міста";
    Layout = "_LayoutFormSm";
}

<form asp-action="Create">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <input type="hidden" asp-for="Id"/>
    <div class="form-group">
        <label asp-for="Name" class="control-label"></label>
        <input asp-for="Name" class="form-control"/>
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Region" class="control-label"></label>
        <input asp-for="Region" class="form-control"/>
        <span asp-validation-for="Region" class="text-danger"></span>
    </div>
    <div class="d-flex">
        <div class="ml-auto px-2">
            <input type="submit" value="Зберегти" class="btn btn-primary"/>
        </div>
        <div>
            <a class="btn btn-secondary " asp-action="Index">Скасувати</a>
        </div>
    </div>
</form>
```

Файл Delete.cshtml

```
@model Infra.DatabaseAdapter.Models.CityModel

@{
    ViewData["Title"] = "Вилучення міста";
    Layout = "_LayoutFormSm";
}

<h5 class="my-4">Ви впевнені, що хочете видалити '@Html.DisplayFor(model => model.Name)'?</h5>
<form asp-action="Delete">
    <input type="hidden" asp-for="Id"/>

    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Name)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Name)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Region)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Region)
        </dd>
    </dl>

    <div class="d-flex">
        <div>
            <input type="submit" value="Видалити" class="btn btn-danger"/>
        </div>
        <div class="ml-auto">
            <a class="btn btn-secondary mr-4 " asp-action="Index">Скасувати</a>
        </div>
    </div>
</form>
```

Файл Edit.cshtml

```
@model Infra.DatabaseAdapter.Models.CityModel

@{
   ViewData["Title"] = "Редагування міст";
   Layout = "_LayoutFormSm";
}


<form asp-action="Edit">
   <div asp-validation-summary="ModelOnly" class="text-danger"></div>
   <input type="hidden" asp-for="Id"/>
   <div class="form-group">
      <label asp-for="Name" class="control-label"></label>
      <input asp-for="Name" class="form-control"/>
      <span asp-validation-for="Name" class="text-danger"></span>
   </div>
   <div class="form-group">
      <label asp-for="Region" class="control-label"></label>
      <input asp-for="Region" class="form-control"/>
      <span asp-validation-for="Region" class="text-danger"></span>
   </div>
   <div class="form-group">
      <label asp-for="CreatedAt" class="control-label"></label>
      <input asp-for="CreatedAt" class="form-control" readonly/>
      <span asp-validation-for="CreatedAt" class="text-danger"></span>
   </div>
   <div class="form-group">
      <label asp-for="UpdatedAt" class="control-label"></label>
      <input asp-for="UpdatedAt" class="form-control" readonly/>
      <span asp-validation-for="UpdatedAt" class="text-danger"></span>
   </div>

   <div class="d-flex">
      <div class="mr-auto">
         <a asp-action="Delete" asp-route-id="@Model.Id" class="btn btn-outline-danger">
            <i class="fa-solid fa-trash-can"></i>
         </a>
      </div>
      <div class="px-2">
         <input type="submit" value="Зберегти" class="btn btn-primary"/>
      </div>
      <div>
         <a class="btn btn-secondary " asp-action="Index">Скасувати</a>
      </div>
   </div>
</form>
```

Файл Index.cshtml

```
@model IEnumerable<Infra.DatabaseAdapter.Models.CityModel>

@{
   ViewData["Title"] = "Список міст";
   Layout = "_Layout";
}

<div class="container-lg">
   <div class="row h-100 d-flex justify-content-center">
      <div class="col-lg-8 col">
         <div class="bg-white rounded shadow-sm p-4 mb-4">
            <div class="row">
               <div class="d-flex pb-4">
                  <div class="p2">
                     <h4 class="mb-1">@ViewData["Title"]</h4>
                  </div>
                  <div class="ml-auto">
                     <a asp-action="Create" class=" text-whity">
```

```
                    Додати місто
                    <i class="fas fa-plus btn btn-primary ml-1"></i>
                  </a>
                </div>
              </div>
            </div>
            <!-- Subject List -->
            <div class="row">

              <table class="pretty-table table  table-striped">
                <thead>
                <tr>
                  <th>
                    @Html.DisplayNameFor(model => model.Name)
                  </th>
                  <th>
                    @Html.DisplayNameFor(model => model.Region)
                  </th>
                  <th>
                    @Html.DisplayNameFor(model => model.CreatedAt)
                  </th>
                  <th></th>
                </tr>
                </thead>
                <tbody>
                @foreach (var item in Model)
                {
                  <tr>
                    <td>
                      @Html.DisplayFor(modelItem => item.Name)
                    </td>
                    <td>
                      @Html.DisplayFor(modelItem => item.Region)
                    </td>
                    <td>
                      @Html.DisplayFor(modelItem => item.CreatedAt)
                    </td>
                    <td class="text-right">
                      <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-outline-warning">
                        <i class="fa-solid fa-pen-to-square"></i>
                      </a>
                    </td>
                  </tr>
                }
                </tbody>
              </table>
            </div>
            <!-- End Subject List -->
          </div>
        </div>
      </div>
    </div>
```

Сторінки заняття

Файл Create.cshtml

```
@model Web.Models.Lessons.LessonVm

@{
    ViewData["Title"] = "Додати нову зустріч";
    Layout = "_LayoutFormSm";
}


<form asp-action="Create">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
```

```
    <input asp-for="Lesson.TutorId" hidden/>
    <div class="form-group">
        <label asp-for="Lesson.TutorName" class="control-label"></label>
        <input asp-for="Lesson.TutorName" class="form-control" readonly></input>
        <span asp-validation-for="Lesson.TutorId" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Lesson.From" class="control-label"></label>
        <input asp-for="Lesson.From" class="form-control"/>
        <span asp-validation-for="Lesson.From" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Lesson.To" class="control-label"></label>
        <input asp-for="Lesson.To" class="form-control"/>
        <span asp-validation-for="Lesson.To" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Lesson.Comment" class="control-label"></label>
        <input asp-for="Lesson.Comment" class="form-control"/>
        <span asp-validation-for="Lesson.Comment" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Lesson.SubjectId" class="form-check-label mt-2"></label>
        <select asp-for="Lesson.SubjectId" class="form-select" asp-items="@Model.Subjects"></select>
        <span asp-validation-for="Lesson.SubjectId" class="text-danger"></span>
    </div>

    <div class="form-group">
        <div class="input-group">
            <label asp-for="Lesson.StudentsIds" class="form-check-label mt-2"></label>
            <select asp-for="Lesson.StudentsIds" asp-items="@Model.HisStudents" class="selectpicker  form-control" data-
width="100%" data-live-search="true" multiple></select>
            <span asp-validation-for="Lesson.StudentsIds" class="text-danger"></span>
        </div>
    </div>
    <div class="d-flex">
        <div class="ml-auto px-2">
            <input type="submit" value="Зберегти" class="btn btn-primary"/>
        </div>
        <div>
            <a class="btn btn-secondary " asp-action="Index">Скасувати</a>
        </div>
    </div>
</form>
```

Файл Delete.cshtml

```
@model Web.Models.Lessons.LessonVm

@{
    ViewData["Title"] = "Видалення уроку";
    Layout = "_LayoutFormSm";
}

<h5 class="my-4">Ви впевнені, що хочете видалити урок #'@Html.DisplayFor(model => model.Lesson.Id)'?</h5>
<form asp-action="Delete">
    <input type="hidden" asp-for="Lesson.Id" name="id"/>

    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Lesson.SubjectName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Lesson.SubjectName)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Lesson.StudentNames)
```

```
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Lesson.StudentNames)
        </dd>
    </dl>

    <div class="d-flex">
        <div>
            <input type="submit" value="Видалити" class="btn btn-danger"/>
        </div>
        <div class="ml-auto">
            <a class="btn btn-secondary mr-4 " asp-action="Index">Скасувати</a>
        </div>
    </div>
</form>
```

Файл Details.cshtml

```
@model Web.Models.Lessons.LessonVm

@{
    ViewData["Title"] = "Урок #" + Model.Lesson.Id;
    Layout = "_LayoutFormSm";
}

<form asp-action="Details">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <input type="hidden" asp-for="Lesson.Id"/>
    <div class="form-group">
        <label asp-for="Lesson.TutorName" class="control-label"></label>
        <input asp-for="Lesson.TutorName" class="form-control" readonly></input>
    </div>
    <div class="form-group">
        <label asp-for="Lesson.From" class="control-label"></label>
        <input asp-for="Lesson.From" class="form-control" readonly/>
    </div>
    <div class="form-group">
        <label asp-for="Lesson.To" class="control-label"></label>
        <input asp-for="Lesson.To" class="form-control" readonly/>
    </div>
    <div class="form-group">
        <label asp-for="Lesson.Comment" class="control-label"></label>
        <input asp-for="Lesson.Comment" class="form-control" readonly/>
    </div>
    <div class="form-group">
        <label asp-for="Lesson.SubjectName" class="form-check-label mt-2"></label>
        <input asp-for="Lesson.SubjectName" class="form-control" readonly></input>
    </div>

    <div class="form-group">
        <label asp-for="Lesson.StudentNames" class="control-label"></label>
        <select asp-for="Lesson.StudentsIds" class="form-control" asp-items="@Model.HisStudents" readonly></select>
    </div>

    <div class="d-flex">
        @if (Model.IsTutor)
        {
            <div class="pr-2">
                <a asp-action="Delete" asp-route-id="@Model.Lesson.Id" class="btn btn-outline-danger">
                    <i class="fa-solid fa-trash-can"></i>
                </a>
            </div>
        }
        <div class="ml-auto">
            <a class="btn btn-secondary " asp-action="Index">Скасувати</a>
        </div>
    </div>
```

```
</form>
```

Файл Index.cshtml

```
@using Web.Helpers
@model Web.Models.Lessons.ListLessonVm

@{
    ViewData["Title"] = "Уроки";
    Layout = "_LayoutFilterTable";
}

<!-- Head button -->

<div class="d-flex mb-3">
    <div class="p2">
        <h4 class="mb-1">Уроки</h4>
    </div>
    @if (Model.IsTutor)
    {
        <div class="ml-auto">
            <a asp-action="Create" class=" text-whity">
                Додати заняття
                <i class="fas fa-plus btn btn-primary ml-1"></i>
            </a>
        </div>
    }
</div>
<!-- End Head button -->
<!-- Lesson table -->
<table class=" table pretty-table">
    <thead>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Lessons.First().SubjectName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Lessons.First().TutorName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Lessons.First().StudentNames)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Lessons.First().LessonDate)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Lessons.First().From)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Lessons.First().To)
        </th>
        <th nowrap="nowrap">
            Дії
        </th>
    </tr>
    </thead>
    <tbody>
    @foreach (var item in Model.Lessons)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.SubjectName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TutorName)
            </td>
            <td>
```

```
                @item.StudentNames.Truncate(15)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.LessonDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.From)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.To)
            </td>
            <td class="text-right text-nowrap">
                <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-outline-primary">
                    <i class="fa-solid fa-eye"></i>
                </a>
            </td>
        </tr>
    }
    </tbody>
</table>
<!-- End Lesson table -->
```

Сторінки реквесту до заняття

Файл Details.cshtml

```
@using Infra.DatabaseAdapter.Helpers
@using Microsoft.AspNetCore.Mvc.TagHelpers
@using Microsoft.OpenApi.Extensions
@model Domain.Models.LessonRequest

@{
    ViewData["Title"] = "Запит #" + Model.Id;
    Layout = "_LayoutFormSm";
}

<form asp-action="Update">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <input type="hidden" asp-for="Id"/>
    <input type="hidden" asp-for="TutorId"/>
    <div class="form-group">
        <label asp-for="LessonDate" class="control-label"></label>
        <input asp-for="LessonDate" class="form-control" readonly></input>
    </div>
    <div class="form-group">
        <label asp-for="Status" class="form-check-label mt-2"></label>
        <input asp-for="Status" class="form-control" readonly></input>
    </div>
    <div class="form-group">
        <label asp-for="Subject" class="control-label"></label>
        <input asp-for="Subject" class="form-control" readonly/>
    </div>
    <div class="form-group">
        <label asp-for="TutorName" class="control-label"></label>
        <input asp-for="TutorName" class="form-control" readonly/>
    </div>
    <div class="form-group">
        <label asp-for="UserName" class="control-label"></label>
        <input asp-for="UserName" class="form-control" readonly/>
    </div>
    <div class="form-group">
        <label asp-for="From" class="control-label"></label>
        <input asp-for="From" class="form-control" readonly/>
    </div>
    <div class="form-group">
        <label asp-for="To" class="control-label"></label>
        <input asp-for="To" class="form-control" readonly/>
```

```
    </div>
    <div class="form-group">
        <label asp-for="Comment" class="control-label"></label>
        <textarea asp-for="Comment" class="form-control nborder" rows="3" readonly></textarea>
    </div>
    <div class="form-group">
        <label asp-for="TutorComment" class="control-label"></label>
        @if (Model.Status == LessonRequestStatus.New)
        {
            <textarea asp-for="TutorComment" rows="6" id="tutorComment" class="form-control">        </textarea>
        }
        else
        {
            <a href="@Model.TutorComment" target="_blank" class="form-control">@Model.TutorComment</a>
        }
        <span asp-validation-for="TutorComment" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Status" class="form-check-label mt-2"></label>
        <div class="list-group list-group-flush">
            <div class="input-group">
                @if (Model.Status == LessonRequestStatus.New)
                {
                    <select asp-for="Status" class="form-select" asp-
items="Html.GetEnumSelectList<LessonRequestStatus>()">
                    </select>
                }
                else
                {
                    <input value="@Html.DisplayFor(model => model.Status)" class="form-control" readonly/>
                }
            </div>
        </div>
    </div>
    <div class="d-flex">

        <div class="px-2">
            <input type="submit" value="Зберегти" class="btn btn-primary"/>
        </div>
        <div class="ml-auto">
            <a class="btn btn-secondary " asp-action="Index">До списку</a>
        </div>
    </div>
</form>
```

Файл Index.cshtml

```
@using Infra.DatabaseAdapter.Helpers
@using Microsoft.AspNetCore.Mvc.TagHelpers
@model Web.Models.LessonRequest.LessonRequestVm;

@{
    ViewData["Title"] = "Історія запитів";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<div class="tab-pane fade active show" id="schedule" role="tabpanel" aria-labelledby="pills-reviews-tab">
    <div class="bg-white rounded shadow-sm p-4 mb-4">
        <div class="container mt-4">
            <div class="d-flex">
                <div class="p2">
                    <h4 class="mb-1">Історія запитів користувача</h4>
                </div>
            </div>
            <div class="row">
                <partial name="RequestTable" model="@Model.MyRequests"/>
            </div>
```

```
        </div>
    </div>
</div>
<div class="tab-pane fade active show" id="schedule" role="tabpanel" aria-labelledby="pills-reviews-tab">
    <div class="bg-white rounded shadow-sm p-4 mb-4">
        <div class="container mt-4">
            <div class="d-flex">
                <div class="p2">
                    <h4 class="mb-1">Запити користувачів на заняття</h4>
                </div>
            </div>
            <div class="row">
                <partial name="RequestTable" model="@Model.RequestsForMe"/>
            </div>
        </div>
    </div>
</div>

@section Scripts
{

}
```

Файл RequestTable.cshtml

```
@using Infra.DatabaseAdapter.Helpers
@using NuGet.Protocol
@using System.Globalization
@using Web.Helpers
@model IEnumerable<Domain.Models.LessonRequest>
<table class="pretty-table">
    <thead>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Id)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Subject)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.UserName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TutorName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.LessonDate)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.From)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.To)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Status)
        </th>
        @if (Model.Count() > 0 && !Model.First().IsTutor)
        {
            <th>
                @Html.DisplayNameFor(model => model.TutorComment)
            </th>
        }
        <th>
            Доступні дії
        </th>
    </tr>
```

```
    </thead>
    <tbody>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Id)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Subject)
            </td>
            <td>
                @item.UserName.Truncate(7)
            </td>
            <td>
                @item.TutorName.Truncate(7)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.LessonDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.From)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.To)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Status)
            </td>
            @if (!item.IsTutor)
            {
                <td>
                    <a href="@item.TutorComment" target="_blank">@Html.DisplayFor(modelItem =>
item.TutorComment)</a>
                </td>
            }
            <td>
                @if (item.IsTutor)
                {
                    <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-outline-primary">
                        <i class="fa-solid fa-eye"></i>
                    </a>
                }
            </td>
        </tr>
    }
    </tbody>
</table>
```

Сторінки профілю

Файл _TutorCard.cshtml

```
@model Web.Models.Profile.TutorCardVm
<div class="tab-pane fade active show mb-4" role="tabpanel" aria-labelledby="pills-reviews-tab">
    <div class="bg-white rounded shadow-sm">
        <div class="container-md px-0">
            <div class="row">
                <div class="col col-auto pr-0 col-12 col-md-auto">
                    <img class="img-fluid rounded img-load-control wh-100" src="@Model.UserVm.Avatar" id="avatar"
alt="..."/>
                </div>
                <div class="col m-2">
                    @if (Model.TutorVm != null)
                    {
                        <span class="star-rating float-right">
```

```
                    <input asp-for="TutorVm!.ReviewValue" class="rating" min="1" max="10" step="1" data-size="md"
data-show-clear="false" data-show-caption="false">
                </span>
            }
            <div class="h4">
                <a class="stretched-link" asp-controller="Profile" asp-action="Details" asp-route-
id="@Model.UserVm.Id">
                    @Model.UserVm.FullName
                </a>
            </div>
            @if (Model.TutorVm != null)
            {
                <div class="d-flex h-50 w-100">
                    <span class="">
                        @Model.TutorVm.Descriptions
                    </span>
                </div>
            }
            <div class="">
                <div class="ml-auto text-light bg-primary rounded-2 px-3 float-right">
                    @Model.TutorVm?.HourRate грн/час
                </div>
                <small class="text-muted align-bottom">
                    <i class="fa-solid fa-location-dot"></i> @Model.UserVm.CityName
                </small>
            </div>
        </div>
    </div>

    @if (Model.TutorVm != null)
    {
        <div class="row mx-0 w-100 card-header py-1">
            <div class="col-auto"><i class="fa-solid fa-comment-dots"></i> @Model.TutorVm.ReviewCount відгуків
 <i class="fa-solid fa-people-line"></i> @Model.TutorVm.LessonCount занять </div>
            <div class="col">
                @for (var i = 0; i < Model.TutorVm.SubjectIds.Count; i++)
                {
                    if (i < 6)
                    {
                        <div class="badge bg-secondary"> @Model.Subjects[Model.TutorVm.SubjectIds[i]].Text</div>
                    }
                    else
                    {
                        <div class="badge bg-gray-400">
                            <i class="fa-solid fa-circle-dot"></i>
                             
                            <i class="fa-solid fa-circle-dot"></i>
                             
                            <i class="fa-solid fa-circle-dot"></i>
                        </div>
                        break;
                    }
                }
            </div>
        </div>
    }
</div>

</div>
</div>
```

Файл Details.cshtml

```
@model Web.Models.Profile.ProfileVm

@{
    ViewBag.Title = Model.UserVm.FullName;
```

```
    Layout = "_Layout";
}


<!-- Profile header-->

<div class="d-sm-flex align-items-end justify-content-end  mb-4">

    @if (@Model.IdentityId == Model.UserVm.Id)
    {
        <a asp-action="Edit" class="d-none d-sm-inline-block btn btn-sm btn-primary shadow-sm">
            <i
                class="fas fa-user fa-sm text-white-50 pr-2">
            </i> Оновити профіль
        </a>
    }
    else
    {
        <input type="checkbox" class="btn-check" id="btn-favorite" autocomplete="off"
               data-userid="@Model.IdentityId" data-tutorid="@Model.UserVm.Id">
        <label class="d-none d-sm-inline-block btn btn-sm btn-outline-pink shadow-sm" for="btn-favorite">
            <i class="fa-solid fa-heart"></i> Улюблені репетитори
        </label>
    }
</div>
<!-- End Profile header -->

<!-- User card -->
<partial name="_TutorCard" model="Model"/>
<!-- End User card -->

@if (@Model.TutorVm != null)
{
    <!-- Anchor block-->
    <div class="container d-flex justify-content-center sticky-top mb-4">
        <div class="btn-group" role="group">
            <button type="button" class="btn btn-outline-primary" onclick="window.location='#subjects';"
checked>Предмети</button>
            <button type="button" class="btn btn-outline-primary" onclick="window.location='#about_tutor';">Про
себе</button>
            <button type="button" class="btn btn-outline-primary"
onclick="window.location='#schedule';">Календар</button>
            <button type="button" class="btn btn-outline-primary" onclick="window.location='#reviews';">Відгуки</button>
        </div>
    </div>
    <!-- Anchor block-->
}

@if (@Model.TutorVm != null)
{
    <!-- Subjects -->
    <div class="tab-pane fade active show" id="subjects" role="tabpanel" aria-labelledby="pills-reviews-tab">
        <div class="bg-white rounded shadow-sm p-4 mb-4">
            <div class="container mt-4 h4-l h3">
                <h4 class="mb-3 text-center">Предмети</h4>
                <div class="container">
                    @for (var i = 0; i < Model.TutorVm.SubjectIds.Count; i++)
                    {
                        <span class="badge bg-light text-black-50 m-2">
@Model.Subjects[Model.TutorVm.SubjectIds[i]].Text</span>
                    }
                </div>
            </div>
        </div>
    </div>

    <!-- End Subjects -->
```

```
<!-- About tutor -->
@if (Model.TutorVm.About.Length > 50)
{
    <div class="tab-pane fade active show" id="about_tutor" role="tabpanel" aria-labelledby="pills-reviews-tab">
        <div class="bg-white rounded shadow-sm p-4 mb-4">
            <div class="container mt-4">
                <h4 class="mb-3 text-center">Про себе</h4>
                <p class="text-wrap text-break w-100">
                    <span style="white-space: pre-line">@Model.TutorVm.About</span>
                </p>
            </div>
        </div>
    </div>
}

<!-- End About tutor -->

<!-- Places -->
<div class="tab-pane fade active show" role="tabpanel" aria-labelledby="pills-reviews-tab">
    <div class="bg-white rounded shadow-sm p-4 mb-4">
        <div class="container mt-4">
            <!-- Type of lesson -->
            <div class="row row-cols-auto">
                <div class="col col-lg-3 col-12">
                    <h4 class="mb-3">Типи занять</h4>
                </div>
                <div class="col">
                    <div href="#" class='btn btn-icon-split mx-auto @(@Model.TutorVm.OnlineAccess ? "btn-info" : "btn-
secondary")'>
                        <span class="icon text-white-50">
                            <i class='fas @(@Model.TutorVm.OnlineAccess ? "fa-check" : "fa-x")'></i>
                        </span>
                        <span class="text">Онлайн</span>
                    </div>
                </div>
                <div class="col">
                    <div href="#" class='btn btn-icon-split mx-auto @(@Model.TutorVm.StudentHomeAccess ? "btn-info" :
"btn-secondary")'>
                        <span class="icon text-white-50">
                            <i class='fas @(@Model.TutorVm.StudentHomeAccess ? "fa-check" : "fa-x")'></i>
                        </span>
                        <span class="text">В Учня</span>
                    </div>
                </div>
                <div class="col">
                    <div href="#" class='btn btn-icon-split mx-auto @(@Model.TutorVm.TutorHomeAccess ? "btn-info" :
"btn-secondary")'>
                        <span class="icon text-white-50">
                            <i class='fas @(@Model.TutorVm.TutorHomeAccess ? "fa-check" : "fa-x")'></i>
                        </span>
                        <span class="text">У вчителя</span>
                    </div>
                </div>
                <div class="col">
                </div>
            </div>
            <!-- End Type of lesson -->
            <div class="row row-cols-auto  mt-5">
                <div class="col col-lg-3">
                    <h5 class="card-title">Місто</h5>
                </div>
                <div class="col col-lg-9 col-md-12">
                    <span style="white-space: pre-line">@Model.Cities.FirstOrDefault(x => x.Value ==
Model.UserVm.CityId)?.Text</span>
                </div>
            </div>
```

```
            <!-- Address -->
            <div class="row row-cols-auto  mt-5">
              <div class="col col-lg-3">
                <h5 class="card-title">Домашня адреса</h5>
              </div>
              <div class="col col-lg-9 col-md-12">
                <span style="white-space: pre-line">@Model.TutorVm.Address</span>
              </div>
            </div>
            <!-- End Address -->
          </div>
        </div>
      </div>
      <!-- End Places -->

      <!-- Calendar -->
      <div class="tab-pane fade active show" id="schedule" role="tabpanel" aria-labelledby="pills-reviews-tab">
        <div class="bg-white rounded shadow-sm p-4 mb-4">
          <div class="container mt-4">
            <partial name="RequestForm" model="@Model.CreateRequestCommand"/>
          </div>
        </div>
      </div>
      <!-- End Calendar -->
      <!-- Reviews -->
      <div class="tab-pane fade active show" id="reviews" role="tabpanel" aria-labelledby="pills-reviews-tab">
        <div class="bg-white rounded shadow-sm p-4 mb-4 restaurant-detailed-ratings-and-reviews">
          <div class="row row-cols-lg-2">
            <div class="col">
              <h4 class="mb-1">Усі рейтинги та огляди</h4>
            </div>
            <div class="col d-flex justify-content-end">
              <a asp-controller="Review" asp-action="Edit" asp-route-tutorId="@Model.TutorVm.Id"
                class="btn btn-warning btn-icon-split">
                <span class="icon text-white-50">
                  <i class="fas fa-star"></i>
                </span>
                <span class="text">
                  Залишити відгук
                </span>
              </a>
            </div>
          </div>
          @foreach (var review in Model.Reviews)
          {
            <partial name="ReviewBlock" model="@review"/>
          }
          <!-- Without reviews -->
          @if (Model.Reviews.Count == 0)
          {
            <h5 class="mt-5">Репетитор ще не має відгуків, але ви можете додати свій</h5>
          }
        </div>
      </div>
      <!-- End Reviews -->
      @section Scripts
      {
        <script type="text/javascript" src="~/js/details-calendar.js" asp-append-version="true"></script>
      }
}
```

Файл Edit.cshtml

```
@model Web.Models.Profile.ProfileVm

@{
    ViewBag.Title = "Редагувати профіль";
```

```
    Layout = "_Layout";
}


<partial name="EditUser" model="Model"/>


@if (Model.UserVm.ProfileEnabled && Model.TutorVm != null)
{
    <!-- Calendar -->
    <div class="tab-pane fade active show" id="schedule" role="tabpanel" aria-labelledby="pills-reviews-tab">
        <div class="bg-white rounded shadow-sm p-4 mb-4">
            <div class="container mt-4">
                <div class="d-flex">
                    <div class="p2">
                        <h4 class="mb-1">Графік свободного часу репетитора</h4>
                        <div><i class="fa fa-info "></i> Зображені години, у які є можливість займатися з новими
учнями</div>
                    </div>
                </div>
                <div class="row d-flex justify-content-end">
                    <div id="calendar" data-tutorid="@Model.TutorVm.Id"></div>
                </div>
            </div>
        </div>
    </div>
    <!-- End Calendar -->
    @section Scripts
    {
        <script type="text/javascript" src="~/js/edit-calendar.js" asp-append-version="true"></script>
    }

}
```

Файл EditUser.cshtml

```
@model Web.Models.Profile.ProfileVm

<form asp-action="Edit" method="post" enctype="multipart/form-data">
    <div class="bg-white rounded shadow-sm p-4 mb-4">
        <div class="container mt-4">
            <div class="d-flex">
                <div class="p2">
                    <h4 class="mb-1">@ViewData["Title"]</h4>
                </div>
            </div>
            <!--User Data-->
            <div class="row row-cols-md-2">
                <div asp-validation-summary="ModelOnly" class="text-danger"></div>
                <input type="hidden" asp-for="UserVm.Id"/>
                <div class="form-group">
                    <label asp-for="UserVm.Name" class="form-check-label mt-2"></label>
                    <textarea class="form-control" rows="1" asp-for="UserVm.Name"></textarea>
                    <span asp-validation-for="UserVm.Name" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="UserVm.Surname" class="form-check-label mt-2"></label>
                    <textarea class="form-control" rows="1" asp-for="UserVm.Surname"></textarea>
                    <span asp-validation-for="UserVm.Surname" class="text-danger"></span>
                </div>
                <div class="form-group">

                    <label asp-for="UserVm.Patronymic" class="form-check-label mt-2"></label>
                    <textarea class="form-control" rows="1" asp-for="UserVm.Patronymic" id="-text"></textarea>
                    <span asp-validation-for="UserVm.Patronymic" class="text-danger"></span>
                </div>
                <div class="form-group">
```

```
                <label asp-for="UserVm.CityId" class="form-check-label mt-2"></label>
                <select asp-for="UserVm.CityId" class="form-select" asp-items="@Model.Cities"></select>
                <span asp-validation-for="UserVm.CityId" class="text-danger"></span>
            </div>

            <div class="form-group">
                <label asp-for="NewAvatar" class="form-check-label mt-2"></label>
                <input asp-for="NewAvatar" type="file" class="form-control" accept=".png, .jpg, .jpeg, .gif, .webp"/>
                <span asp-validation-for="NewAvatar" class="text-danger"></span>
            </div>

        </div>
    </div>
    <!-- End User Data-->
</div>
<!--Tutor Profile-->
@if (Model.TutorVm != null && Model.UserVm.ProfileEnabled)
{
    <div class="bg-white rounded shadow-sm p-4 mb-4">
        <div class="container mt-4">
            <div class="d-flex">
                <div class="p2">
                    <h4 class="mb-2">@ViewData["Title"]</h4>
                </div>
            </div>
            <div class="row">
                <div asp-validation-summary="ModelOnly" class="text-danger"></div>
                <input type="hidden" asp-for="@Model.TutorVm.Id"/>
                <div class="form-group mb-2">
                    <div class="list-group">
                        <label asp-for="@Model.TutorVm.SubjectIds" class="control-label"></label>
                        <div class="input-group">
                            <select asp-for="@Model.TutorVm.SubjectIds" asp-items="@Model.Subjects" class="selectpicker
form-control" data-width="100%" data-live-search="true" multiple></select>
                        </div>
                    </div>
                </div>
                <row class="row row-cols-lg-2 row-cols-1 pr-0 mt-2">
                    <div class="form-group mb-2 col col-md-8">
                        <label class="control-label">Місце проведення занять</label>
                        <br/>
                        <div class="form-check form-check-inline form-switch">
                            <input class="form-check-input" type="checkbox" asp-for="@Model.TutorVm.TutorHomeAccess">
                            <label asp-for="@Model.TutorVm.TutorHomeAccess" class="form-check-label"></label>
                        </div>
                        <div class="form-check form-check-inline form-switch">
                            <input class="form-check-input" type="checkbox" asp-
for="@Model.TutorVm.StudentHomeAccess">
                            <label asp-for="@Model.TutorVm.StudentHomeAccess" class="form-check-label"></label>
                        </div>
                        <div class="form-check form-check-inline form-switch">
                            <input class="form-check-input" type="checkbox" asp-for="@Model.TutorVm.OnlineAccess">
                            <label asp-for="@Model.TutorVm.OnlineAccess" class="form-check-label"></label>
                        </div>
                    </div>
                    <div class="form-group col col-md-4 pr-0">
                        <label asp-for="@Model.TutorVm.HourRate" class="control-label"></label>
                        <input asp-for="@Model.TutorVm.HourRate" class="form-control"/>
                        <span asp-validation-for="@Model.TutorVm.HourRate" class="text-danger"></span>
                    </div>
                </row>

                <div class="form-group">
                    <label asp-for="@Model.TutorVm.Address" class="form-check-label mt-2"></label>
                    <textarea class="form-control" rows="2" asp-for="@Model.TutorVm.Address" id="address-
text"></textarea>
```

```
                <span asp-validation-for="@Model.TutorVm.Address" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="@Model.TutorVm.Descriptions" class="form-check-label  mt-2"></label>
                <textarea class="form-control" rows="4" asp-for="@Model.TutorVm.Descriptions"></textarea>
                <span asp-validation-for="@Model.TutorVm.Descriptions" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="@Model.TutorVm.About" class="form-check-label  mt-2"></label>
                <textarea class="form-control" rows="16" asp-for="@Model.TutorVm.About"></textarea>
                <span asp-validation-for="@Model.TutorVm.About" class="text-danger"></span>
            </div>

        </div>
    </div>
</div>
}
<!--End Tutor Profile-->
<div class="d-sm-flex align-items-end justify-content-end mb-4">

    <div class="form-check form-switch pr-md-5">
        <input class="form-check-input" type="checkbox" asp-for="UserVm.ProfileEnabled">
        <label class="form-check-label" asp-for="UserVm.ProfileEnabled"></label>
    </div>
    <button type="submit" class="d-none d-sm-inline-block btn btn-sm btn-primary shadow-sm">
        <i class="fas fa-floppy-disk text-white-50 pr-2"></i> Зберегти зміни в анкеті
    </button>
</div>
</form>
```

Файл Index.cshtml

```
@using Web.Models.Shared
@using Microsoft.AspNetCore.Mvc.TagHelpers
@model Web.Models.Profile.SearchVm

@{
    ViewBag.Title = "Пошук репетитора";
    Layout = "_Layout";
}

<form method="get">
    <!-- Search Input -->
    <div class="row mb-5 d-flex justify-content-center">
        <div class="input-group">
            <input asp-for="Filters.SearchText" class="form-control border-left-primary" placeholder="Пошук ..."
                   aria-label="Search" aria-describedby="basic-addon2">
            <div class="input-group-append">
                <button class="btn btn-primary" type="submit">
                    <i class="fa-solid fa-magnifying-glass"></i>
                </button>
            </div>
        </div>
    </div>
    <!-- End Search Input -->
    <div class="row">
        <div class="col col-lg-3 col-12">
            <!-- Side - Rate -->
            <div class="card shadow mb-4">
                <div class="card-header py-3 d-flex flex-row align-items-center justify-content-between">
                    <h6 class="m-0 font-weight-bold text-primary">Вартість за годину</h6>
                </div>
                <div class="list-wrap open p-2" style="">
                    <div class="input-group mb-3">
                        <div class="input-group-prepend">
                            <span class="input-group-text">Від </span>
                        </div>
```

```
                    <input asp-for="Filters.HourRateFrom" min="0" type="number" class="form-control" placeholder="0
грн" oninvalid="this.setCustomValidity('Значення має бути більше або рівне 0')">
                    <div class="input-group-append">
                        <span class="input-group-text">грн</span>
                    </div>
                </div>
                <div class="input-group mb-3">
                    <div class="input-group-prepend">
                        <span class="input-group-text">До </span>
                    </div>
                    <input asp-for="Filters.HourRateTo" min="0" type="number" class="form-control" placeholder="0 грн"
oninvalid="this.setCustomValidity('Значення має бути більше або рівне 0')">
                    <div class="input-group-append">
                        <span class="input-group-text">грн</span>
                    </div>
                </div>
            </div>
        </div>

        <div class="card shadow mb-4">
            <div class="card-header py-3 d-flex flex-row align-items-center justify-content-between">
                <h6 class="m-0 font-weight-bold text-primary">Місто</h6>
            </div>
            <div class="list-wrap open p-2" style="">
                <select asp-for="Filters.CityId" asp-items="@Model.Cities" class="selectpicker  form-control" data-live-
search="true"></select>
            </div>
        </div>


        <div class="card shadow mb-4">
            <div class="card-header py-3 d-flex flex-row align-items-center justify-content-between">
                <h6 class="m-0 font-weight-bold text-primary">Предмет за годину</h6>
            </div>
            <div class="list-wrap open p-2" style="">
                <select asp-for="Filters.SubjectId" asp-items="@Model.Subjects" class="selectpicker  form-control" data-
live-search="true"></select>
            </div>
        </div>

        <div class="card shadow mb-4">
            <div class="card-header py-3 d-flex flex-row align-items-center justify-content-between">
                <h6 class="m-0 font-weight-bold text-primary">Тип заняття</h6>
            </div>
            <div class="list-wrap open p-2 ml-3" style="">

                <div class="form-check">
                    <input asp-for="Filters.OnlineAccess" class="form-check-input" type="checkbox" id="check1">
                    <label class="form-check-label" for="check1">Онлайн</label>
                </div>
                <div class="form-check">
                    <input asp-for="Filters.TutorHomeAccess" class="form-check-input" type="checkbox" id="check2">
                    <label class="form-check-label" for="check2">У вчителя</label>
                </div>
                <div class="form-check">
                    <input asp-for="Filters.StudentHomeAccess" class="form-check-input" type="checkbox" id="check3">
                    <label class="form-check-label" for="check3">У студента</label>
                </div>
            </div>
        </div>
        <!-- Favorite -->

        <div class="card shadow mb-4">
            <input asp-for="Filters.OnlyFavorites" type="checkbox" class="btn-check" id="only-favorite">
            <label class="d-none d-sm-inline-block btn btn-sm btn-outline-pink shadow-sm mb-0" for="only-favorite">
                <i class="fa-solid fa-heart"></i> Тільки улюблені репетитори
            </label>
```

```
            </div>
            <!-- Submit -->
            <div class="card shadow mb-4">

                <button class="btn btn-primary" type="submit">
                    <i class="fa-solid fa-magnifying-glass"></i> Шукати
                </button>
            </div>


        </div>
        <div class="col col-lg-9 col-md-12">
            <!-- TutorCard -->
            @foreach (var x in Model.TutorCards)
            {
                <partial name="_TutorCard" model="@x"/>
            }
            <!-- End TutorCard -->
        </div>
    </div>
</form>
```

Файл RequestForm.cshtml

```
@model Domain.Commands.CreateRequestCommand

<div class="d-flex">
    <div class="p2">
        <h4 class="mb-1">Графік свободного часу репетитора</h4>
        <div><i class="fa fa-info "></i> Зображені години, у які є можливість займатися з новими учнями</div>
    </div>
    <div class="ml-auto">
        <a onclick="addNewEvent(null);" class=" text-whity">
            Додати заявку на заняття
            <i class="fas fa-plus btn btn-primary ml-1"></i>
        </a>
    </div>
</div>
<div class="row d-flex justify-content-end">
    <div id="calendar" data-tutorid="@Model.TutorId" data-userid="@Model.CreatedId"></div>
</div>

<div class="modal fade" id="requestModal" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-
labelledby="requestModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <form asp-controller="LessonRequest" asp-action="Create" onsubmit="send_request(this);return false;"
name="requestForm">
                <div class="modal-header">
                    <h5 class="modal-title" id="requestModalLabel">Запис до репетитора</h5>
                </div>
                <div class="modal-body mb-3">
                    <input asp-for="TutorId" id="tutorProfileId" class="form-control" type="hidden"/>

                    <div class="list-group list-group-flush">
                        <div class="input-group">
                            <select asp-for="SubjectId" class="form-select" asp-items="@Model.Subjects" id="subject"></select>
                            <span asp-validation-for="SubjectId" class="text-danger"></span>
                        </div>

                        <div class="form-group">
                            <label asp-for="From" class="form-label"></label>
                            <input asp-for="From" id="timeFrom" class="form-control" type="datetime-local"></input>
                        </div>
                        <div class="form-group">
                            <label asp-for="To" class="form-label"></label>
                            <input asp-for="To" id="timeTo" class="form-control" type="datetime-local"></input>
```

```
                </div>
                <div class="form-group">
                    <label asp-for="Comment" class="form-label"></label>
                    <textarea asp-for="Comment" rows="6" id="tutorComment" class="form-control"></textarea>
                    <span asp-validation-for="Comment" class="text-danger"></span>
                </div>
            </div>
        </div>
        <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Скасувати</button>
            <button type="submit" class="btn btn-primary">Підтвердити</button>
        </div>
    </form>
  </div>
 </div>
</div>
<div class="modal fade" id="auth-modal" tabindex="-1" aria-labelledby="authmodal" aria-hidden="true">
 <div class="modal-dialog">
  <div class="modal-content">
   <div class="modal-header">
    <h5 class="modal-title" id="authmodal">Інформаційне повідомлення</h5>
    <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
   </div>
   <div class="modal-body">
    Дорогий гість, зверни увагу, що тільки зареєстровані користувачі можуть надсилати запити до
репетиторів.
    <a class="link-primary" asp-area="Identity" asp-page="/Account/Login">Вхід</a>
   </div>
   <div class="modal-footer">
    <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
   </div>
  </div>
 </div>
</div>
```

Файл ReviewBlock.cshtml

```
@using Microsoft.AspNetCore.Mvc.TagHelpers
@model Domain.Models.Review

<div class="reviews-members pt-4 pb-4">
 <div class="media">
  <img alt="Generic placeholder image" src="@Model.Avatar" class="mr-3 rounded-pill img-load-control">
  <div class="media-body">
   <div class="reviews-members-header">
    <span class="star-rating float-right">
     <input asp-for="Rating" class="rating" min="1" max="10" data-size="md" data-show-clear="false" data-
show-caption="false" disabled>
    </span>
    <h6 class="mb-1">
     <a class="text-black" href="#">@Model.AuthorName</a>
    </h6>
    <p class="text-gray">
     @Model.UpdatedAt
     @if (Model.CreatedAt != Model.UpdatedAt)
     {
      <span class="text-secondary">( Створено: @Model.CreatedAt )</span>
     }
    </p>
   </div>
   <div class="reviews-members-body">
    <p class="text-wrap text-break w-100">@Model.Description</p>
   </div>
   <div class="reviews-members-footer">
   </div>
  </div>
 </div>
```

```
</div>
<hr>
```

Сторінки пов’язані із видаленим користувачем

Файл Index.cshtml

```
@model Web.Models.RemoveUser.RemoveUserVm;

@{
    ViewBag.Title = "Видалення даних користувачів";
    Layout = "_LayoutFormSm";
}

<form asp-action="DeactivateUser" method="post" class="text-center">
    <div class="form-group">
        <label asp-for="UserId" class="form-check-label mt-2"></label>
        <select asp-for="UserId" class="selectpicker  form-control" data-live-search="true" asp-
items="@Model.UserList"></select>
        <span asp-validation-for="UserId" class="text-danger"></span>
    </div>
    <button class="btn btn-danger btn-block" type="submit">Видалити дані користувача</button>
</form>
```

Файл Removed.cshtml

```
@{
    ViewBag.Title = "Ваш обліковий запис деактивовано";
    Layout = "_LayoutFormSm";
}

<span class="py-4">Здається, адміністратор заблокував ваш обліковий запис. Бажаєте видалити свої персональні
дані?</span>
<div class="w-auto">
    <form class="pt-2" asp-action="RemoveMePersonalData" method="post">
        <button type="submit" class="btn btn-danger btn-block"><i class="fa fa-remove"></i> Видалити мої персональні
дані</button>
    </form>
</div>
```

Сторінка відгуку

Файл Edit.cshtml

```
@model  Domain.Models.Review

@{
    ViewData["Title"] = "Ваш відгук";
    Layout = "_LayoutFormSm";
}
<form asp-action="Edit">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <input asp-for="TutorId" hidden/>
    <input asp-for="AuthorId" hidden/>
    <div class="form-group">
        <label asp-for="TutorName" class="control-label"></label>
        <input asp-for="TutorName" class="form-control" readonly/>
    </div>
    <div class="form-group">
        <label asp-for="AuthorName" class="control-label"></label>
        <input asp-for="AuthorName" class="form-control" readonly/>
        <span asp-validation-for="AuthorId" class="text-danger"></span>
    </div>
    <div class="row-group d-flex">
        <div class="ml-auto">
            <input asp-for="Rating" type="number" class="rating ml-auto"
                   min="1" max="10" step="1" data-size="lg" data-show-clear="false" data-show-caption="false">
```

```
            <span asp-validation-for="Rating" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <label asp-for="Description" class="control-label"></label>
        <textarea asp-for="Description" rows="10" class="form-control"></textarea>
        <span asp-validation-for="Description" class="text-danger"></span>
    </div>
    <div class="d-flex">
        <div class="ml-auto pr-2">
            <input type="submit" value="Зберегти" class="btn btn-primary"/>
        </div>
        <div class="">
            <a class="btn btn-secondary" asp-controller="Profile" asp-action="Details" asp-route-
id="@Model.TutorId">Скасувати</a>
        </div>
    </div>
</form>
```

Shared

Файл Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <title>@ViewData["Title"]</title>


    <!-- Custom fonts -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css"/>
    <link
href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i"
rel="stylesheet">

    <!-- bootstrap styles -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-select@1.14.0-beta3/dist/css/bootstrap-
select.min.css">

    <!-- Star rating styles -->
    <link href="https://cdn.jsdelivr.net/gh/kartik-v/bootstrap-star-rating@4.1.2/css/star-rating.min.css" media="all"
rel="stylesheet" type="text/css"/>
    <link href="https://cdn.jsdelivr.net/gh/kartik-v/bootstrap-star-rating@4.1.2/themes/krajee-svg/theme.css" media="all"
rel="stylesheet" type="text/css"/>

    <!-- Tables styles -->
    <link rel="stylesheet" href="https://cdn.datatables.net/2.0.5/css/dataTables.dataTables.min.css">

    <!-- Custom styles -->
    <link href="~/css/sb-admin-2.css" rel="stylesheet">


    @* <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-icons.css"> *@
    <link rel="stylesheet" href="~/css/tutor-card.css" asp-append-version="true"/>
    <link rel="stylesheet" href="~/css/custom.css" asp-append-version="true"/>

</head>

<body>
```

```
<!-- Page Wrapper -->
<div id="wrapper">

    <!-- Content Wrapper -->
    <div id="content-wrapper" class="d-flex flex-column">

        <!-- Main Content -->

        <header>
            <partial name="_NavBar"/>
        </header>
        <div class="container-md mt-4 p-0">
            <div class="col-md-12">
                @RenderBody()
            </div>
        </div>
        <!--  End of Main Content -->

        <!-- Footer -->
        <footer class="sticky-footer bg-white">
            <div class="container my-auto">
                <div class="copyright text-center my-auto">
                    <span>
                        Copyright &copy; copy; 2024 - Web -
                        <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
                    </span>
                </div>
            </div>
        </footer>
        <!-- End of Footer -->

    </div>
    <!-- End of Content Wrapper -->

</div>
<!-- End of Page Wrapper -->

<!-- Scroll to Top Button-->
<a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
</a>

<!-- Logout Modal-->
<div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"
     aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Бажаєте вийти з сайту?</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">Виберіть "Вийти" нижче, якщо ви готові завершити поточний сеанс.</div>
            <div class="modal-footer">
                <form class="form-inline" asp-area="Identity" asp-page="/Account/Logout" asp-route-
returnUrl="@Url.Action("Index", "Profile", new { area = "" })">
                    <button type="submit" class="btn btn-primary">Вийти з сайту</button>
                </form>
            </div>
        </div>
    </div>
</div>


<!-- Bootstrap core JavaScript-->
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap-select@1.14.0-beta3/dist/js/bootstrap-select.min.js"></script>

<!-- Star rating JavaScript -->
<script src="https://cdn.jsdelivr.net/gh/kartik-v/bootstrap-star-rating@4.1.2/js/star-rating.min.js"
type="text/javascript"></script>
<script src="https://cdn.jsdelivr.net/gh/kartik-v/bootstrap-star-rating@4.1.2/themes/krajee-svg/theme.js"></script>
<script src="https://cdn.jsdelivr.net/gh/kartik-v/bootstrap-star-rating@4.1.2/js/locales/ua.js"></script>

<!-- Page level plugins -->

<script src="https://cdn.jsdelivr.net/npm/fullcalendar@6.1.11/index.global.min.js"></script>

<!-- Tables JavaScript -->
<script src="https://cdn.datatables.net/2.0.5/js/dataTables.min.js"></script>

<!-- Custom scripts for all pages-->
<script src="~/vendor/chart.js/Chart.min.js"></script>
<script src="~/js/sb-admin-2.min.js"></script>
<script src="~/js/file.js"></script>
<script src="~/js/main.js"></script>
<script src="~/js/avatar.js"></script>

@await RenderSectionAsync("Scripts", false)
</body>
</html>
```

Файл _LayoutFilterTable.cshtml

```
@{
    Layout = "/Views/Shared/_Layout.cshtml";
}

<div class="tab-pane fade active show" role="tabpanel" aria-labelledby="pills-reviews-tab">
    <div class="bg-white rounded shadow-sm p-4 mb-4">
        <div class="container mt-4">
            @RenderBody()
        </div>
    </div>
</div>

@section Scripts {
    @RenderSection("Scripts", false)
}
```

Файл _LayoutFormSm.cshtml

```
@{
    Layout = "_Layout";
}
<div class="container-sm">
    <div class="row h-100 d-flex align-items-center justify-content-center">
        <div class="col-lg-6">
            <div class="bg-white rounded shadow-sm p-5 mb-4">
                <h4 class="mb-3 text-center">@ViewData["Title"]</h4>
                @RenderBody()
            </div>
        </div>
    </div>
</div>

@section Scripts {
    @RenderSection("Scripts", false)
}
```

Файл _NavBar.cshtml

```
@inject SignInManager<UserModel> SignInManager
@inject UserManager<UserModel> UserManager
<!-- Topbar -->
<nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-top shadow">

    <!-- Topbar Navbar -->
    <a class="navbar-brand" asp-controller="Profile" asp-action="Index">
        <img src="~/img/logo.png" width="30" height="30" alt="">
    </a>
    <ul class="navbar-nav mx-auto mt-2 mt-lg-0">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Profile" asp-action="Index">Пошук репетиторів</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="LessonRequest" asp-action="Index">Запити</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Lesson" asp-action="Index">Уроки</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Assignment" asp-action="Index">Мої завдання</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Solution" asp-action="Index">Завдання для виконання</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Home" asp-action="About">Про нас</a>
        </li>
    </ul>
    <ul class="navbar-nav ml-auto">
        <div class="topbar-divider d-none d-sm-block"></div>

        @if (SignInManager.IsSignedIn(User))
        {
            <partial name="_SignedNavBar"/>
        }
        else
        {
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Register">Зареєструватися</a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Login">Вхід</a>
            </li>
        }
    </ul>

</nav>
<!-- End of Topbar -->
```

Файл _SignedNavBar.cshtml

```
@using System.Security.Claims
@inject SignInManager<UserModel> SignInManager
@inject UserManager<UserModel> UserManager
@{
    ViewBag.Avatar = User.FindFirst(ClaimTypes.NameIdentifier)?.Value + ".png";
}
<!-- Nav Item - User Information -->
<li class="nav-item dropdown no-arrow" style="min-width: 250px;">
    <a class="nav-link dropdown-toggle" href="#" id="userDropdown" role="button"
       data-bs-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
        <span class="mr-2 d-none d-lg-inline text-gray-600 small">@User.Identity?.Name</span>
        <img class="img-profile img-load-control rounded-circle" src="~/avatars/@ViewBag.Avatar" alt="">
    </a>
```

```
    <!-- Dropdown - User Information -->
    <ul class="dropdown-menu dropdown-menu-end shadow animated--grow-in"
      aria-labelledby="userDropdown">
      <a class="dropdown-item" asp-controller="Profile" asp-action="Details" asp-route-id="0">
        <i class="fas fa-user fa-sm fa-fw mr-2 text-gray-400"></i>
        Профіль
      </a>
      <a class="dropdown-item" asp-area="Identity" asp-page="/Account/Manage/Index">
        <i class="fas fa-cogs fa-sm fa-fw mr-2 text-gray-400"></i>
        Налаштування
      </a>
      @if (User.IsInRole("Administrator"))
      {
        <a class="dropdown-item" asp-controller="City" asp-action="Index">
          <i class="fas fa-building fa-sm fa-fw mr-2 text-gray-400"></i>
          Список міст
        </a>
        <a class="dropdown-item" asp-controller="Subject" asp-action="Index">
          <i class="fas fa-list-alt fa-sm fa-fw mr-2 text-gray-400"></i>
          Список предметів
        </a>
        <a class="dropdown-item" asp-controller="RemoveUser" asp-action="Index">
          <i class="fas fa-list-alt fa-sm fa-fw mr-2 text-gray-400"></i>
          Видалення користувачів
        </a>
        <a class="dropdown-item" href="/swagger/index.html">
          <i class="fas  fa-external-link-square fa-sm fa-fw mr-2 text-gray-400"></i>
          Swagger
        </a>
      }

      <div class="dropdown-divider"></div>
      <a class="dropdown-item" type="button" data-bs-toggle="modal" data-bs-target="#logoutModal">
        <i class="fas fa-sign-out-alt fa-sm fa-fw mr-2 text-gray-400"></i>
        Вийти з системи
      </a>
    </ul>
</li>
```

Файл FilesForm.cshtml

```
@using Microsoft.AspNetCore.Mvc.TagHelpers
@model List<Domain.Models.UserFile>

<div class="upload mb-3">
    <h4 class="row mb-2">Список файлів</h4>
    <div id="liveAlertPlaceholder"></div>
    <!-- Отримати всі файли з сервера -->
    @foreach (var item in Model)
    {
        <div class="d-flex m-1">
            <div class=" mr-auto text-truncate pr-3">@item.FileName</div>
            <div class="ml-2">
                <a type="button" class="btn btn-sm btn-outline-primary"
                  asp-controller="File" asp-action="Download" asp-route-id="@item.Id" target="_blank">
                    <i class="fa-solid fa-arrow-down"></i>
                    <!-- Завантажити файл -->
                </a>
            </div>
        </div>
    }
    @if (Model.Count == 0)
    {
        <h5>Файли відсутні</h5>
    }
</div>
```

Сторінки рішення

Файл Edit.cshtml

```
@using Infra.DatabaseAdapter.Helpers
@using Microsoft.AspNetCore.Mvc.TagHelpers
@using Domain.Helpers
@using Microsoft.OpenApi.Extensions
@model Web.Models.Solutions.SolutionVm

@{
    ViewData["Title"] = Model.Solution.SubjectName + " - " + Model.Solution.AssignmentTitle;
    Layout = "_Layout";
}

<div class="container-sm">
    <div class="row h-100 d-flex align-items-center justify-content-center">
        <div class="col-lg-6">
            <div class="bg-white rounded shadow-sm p-5 mb-4">
                <h4 class="mb-1">@Model.Solution.AssignmentTitle</h4>

                <!-- Assignment -->
                <div class="row">
                    <div class="col">
                        <label asp-for="Solution.SubjectName" class="col"></label>
                    </div>
                    <div class="col text-black">@Model.Solution.SubjectName</div>
                </div>
                <div class="row">
                    <div class="col">
                        <label asp-for="Solution.TutorName" class="col"></label>
                    </div>
                    <div class="col text-black">@Model.Solution.TutorName</div>
                </div>
                <div class="row">
                    <div class="col">
                        <label asp-for="Solution.Deadline" class="col"></label>
                    </div>
                    <div class="col text-black">@Model.Solution.Deadline</div>
                </div>
                @if (Model.Solution.Description != null && Model.Solution.Description.Length > 0)
                {
                    <div class="form-group">
                        <h4 class="row">Опис завдання</h4>
                        <p class="row text-wrap text-break w-100">@Model.Solution.Description</p>
                    </div>
                }
                <!-- End Assignment -->

                <!-- Files -->
                @if (Model.Solution.AssignmentFiles.Count > 0)
                {
                    <div class="upload mb-3">
                        <h4 class="row mb-2">Матеріали до заняття</h4>
                        @foreach (var item in Model.Solution.AssignmentFiles)
                        {
                            <div class="d-flex m-1">
                                <div class=" mr-auto text-truncate pr-3">@item.FileName</div>
                                <div class="ml-2">
                                    <a type="button" class="btn btn-sm btn-outline-primary"
                                       asp-controller="File" asp-action="Download" asp-route-id="@item.Id" target="_blank">
                                        <i class="fa-solid fa-arrow-down"></i>
                                        <!-- Завантажити файл -->
                                    </a>
                                </div>
                            </div>
```

```
                }
            </div>
        }
        <!-- End Files -->
    </div>
</div>
</div>

<!-- Solution -->
<div class="container-sm">
    <div class="row h-100 d-flex align-items-center justify-content-center">
        <div class="col-lg-6">
            <div class="bg-white rounded shadow-sm p-5 mb-4">
                <form asp-action="Edit">
                    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
                    <input type="hidden" asp-for="Solution.Id"/>
                    <input type="hidden" asp-for="IsTutor"/>

                    <div class="form-group">
                        <label asp-for="Solution.StudentName" class="control-label"></label>
                        <input asp-for="Solution.StudentName" class="form-control nborder" readonly/>
                        <span asp-validation-for="Solution.StudentName" class="text-danger"></span>
                    </div>
                    <!-- Student Answer -->
                    <div class="form-group">
                        <label asp-for="Solution.Answer" class="control-label"></label>
                        @if (!Model.IsTutor && Model.Solution.Status == SolutionStatus.Todo)
                        {
                            <textarea asp-for="Solution.Answer" rows="4" class="form-control"></textarea>
                            <span asp-validation-for="Solution.Answer" class="text-danger"></span>
                        }
                        else
                        {
                            <textarea asp-for="Solution.Answer" rows="4" class="form-control nborder" readonly></textarea>
                        }
                    </div>
                    <!-- Tutor comment -->
                    <div class="form-group">
                        <label asp-for="Solution.TutorComment" class="control-label"></label>
                        @if (Model.IsTutor && Model.Solution.Status != SolutionStatus.Completed)
                        {
                            <textarea asp-for="Solution.TutorComment" rows="4"  class="form-control"></textarea>
                            <span asp-validation-for="Solution.TutorComment" class="text-danger"></span>
                        }
                        else
                        {
                            if (Model.Solution.TutorComment?.Length > 0)
                            {
                                <input asp-for="Solution.TutorComment" class="form-control nborder" readonly/>
                            }
                        }
                    </div>
                    <!-- Solution Status -->
                    <div class="form-group">
                        <label asp-for="Solution.Status" class="form-check-label mt-2"></label>
                        @if (Model.IsTutor && Model.Solution.Status != SolutionStatus.Completed)
                        {
                            <select asp-for="Solution.Status" class="form-select" id="status" asp-
items="Html.GetEnumSelectList<SolutionStatus>()"></select>
                            <span asp-validation-for="Solution.Status" class="text-danger"></span>
                        }
                        else
                        {
                            <select asp-for="Solution.Status" class="form-select nborder" id="status" asp-
items="Html.GetEnumSelectList<SolutionStatus>()" disabled></select>
                        }
```

```
                </div>

                <!-- Submit -->
                <div class="d-flex">
                    <div class="ml-auto px-2">
                        <input type="submit" value="Зберегти" class="btn btn-primary"/>
                    </div>
                    <div>
                        <a class="btn btn-secondary " asp-action="Index">Скасувати</a>
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>
<!-- End Solution -->
<!-- Solution Files -->
<div class="container-sm">
    <div class="row h-100 d-flex align-items-center justify-content-center">
        <div class="col-lg-6">
            <div class="bg-white rounded shadow-sm p-5 mb-4">
                <partial name="FilesForm" model="@Model.Solution.SolutionFiles"/>
                <form id="uploadform" asp-controller="File" asp-action="Upload" enctype="multipart/form-data">
                    <input type="hidden" class="mt-3" asp-for="Solution.Id" name="SolutionId"/>
                    <input class="form-control form-control-sm" id="uploadform-file" name="FormFile" type="file">
                </form>
            </div>
        </div>
    </div>
</div>
<!-- End Solution Files -->
```

Файл Index.cshtml

```
@using Infra.DatabaseAdapter.Helpers
@using Web.Helpers
@model Web.Models.Solutions.SolutionListVm

@{
    ViewData["Title"] = "Завдання • Для виконання";
    Layout = "_LayoutFilterTable";
}
<!-- Head button -->

<div class="d-flex mb-3">
    <div class="p2">
        <h4 class="mb-1">@ViewData["Title"]</h4>
    </div>
</div>
<!-- End Head button -->
<!-- Head Filter-->
<form asp-action="Index" method="get" class="my-3">

    <div class="row d-flex row-cols-md-6 justify-content-start">
        <input asp-for="Filter.UserId" name="userid" hidden/>
        <div class="col">
            <label asp-for="Filter.Status" class="form-check-label"></label>
            <select asp-for="Filter.Status" asp-items="@Html.GetEnumSelectList<SolutionStatus>()" class="selectpicker
form-control" data-width="100%" data-live-search="true" multiple></select>
        </div>
        <div class="col">
            <label asp-for="Filter.SubjectId" class="form-check-label"></label>
            <select asp-for="Filter.SubjectId" asp-items="@Model.Subjects" class="selectpicker form-control" data-
width="100%" data-live-search="true" multiple></select>
        </div>
        <div class="col">
```

```
            <label asp-for="Filter.TutorId" class="form-check-label"></label>
            <select asp-for="Filter.TutorId" asp-items="@Model.HisTutors" class="selectpicker form-control" data-
width="100%" data-live-search="true" multiple></select>
        </div>
        <div class="col">
            <label asp-for="Filter.StudentId" class="form-check-label"></label>
            <select asp-for="Filter.StudentId" asp-items="@Model.HisStudents" class="selectpicker form-control" data-
width="100%" data-live-search="true" multiple></select>
        </div>
        <div class="col">
            <label asp-for="Filter.AssignmentId" class="form-check-label"></label>
            <select asp-for="Filter.AssignmentId" asp-items="@Model.Assignments" class="selectpicker form-control" data-
width="100%" data-live-search="true" multiple></select>
        </div>
        <div class="col align-self-end">
            <button type="submit" class="btn btn-primary">
                <i class="fa-solid fa-arrows-rotate"></i>
            </button>
        </div>
    </div>
</form>
<!-- End Head Filter-->


<table class="table pretty-table stripe" style="white-space: nowrap;">
    <thead>
    <tr>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Solutions.First().Id)
        </th>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Solutions.First().SubjectName)
        </th>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Solutions.First().AssignmentTitle)
        </th>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Solutions.First().TutorName)
        </th>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Solutions.First().StudentName)
        </th>
        <th scope="col">
            @Html.DisplayNameFor(model => model.Solutions.First().Status)
        </th>
        <th scope="col"></th>
    </tr>
    </thead>
    <tbody>
    @foreach (var item in Model.Solutions)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Id)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.SubjectName)
            </td>
            <td>
                @item.AssignmentTitle.Truncate(25)
            </td>
            <td>
                @item.TutorName.Truncate(15)
            </td>
            <td>
                @item.StudentName.Truncate(15)
            </td>
```

```
            <td>
                @Html.DisplayFor(modelItem => item.Status)
            </td>
            <td>
                <a asp-action="Edit" asp-route-id="@item.Id" class="btn small-text small btn-outline-primary">
                    <i class="small-text small fa-solid fa-pen-to-square"></i>
                </a>
            </td>
        </tr>
    }
    </tbody>
</table>
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ПОШУКУ РЕПЕТИТОРІВ

Програма та методика тестування

КПІ.ІТ-0221.04544.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Анастасія СКАЧКОВА

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є програмне забезпечення для пошуку репетиторів.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox, …);

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

У процесі тестування має бути перевірений наступний функціонал:

Система авторизації користувача:

- реєстрація користувачів;
- авторизація користувачів;
- вихід із акаунту;
- видалення облікових даних користувачів.

Система управління профілями репетиторів:

- пошук репетиторів;
- створення, редагування профілю репетитора;
- перегляд профілю репетитора;
- створення, редагування відгуку про репетитора;
- додавання рейтингу репетитору;
- додавання репетитора в улюблені.

Система занять:

- створення, редагування та видалення заняття;
- перегляд заняття;
- перегляд списку уроків;
- перегляд списку запитів;
- створення відповіді на запит.

Система завдань

- створення, редагування та видалення завдання;

- перегляд списку виставлених завдань;
- перегляд списку своїх завдань.

Система відповідей:

- перевірка відповідей користувачів;
- створення, редагування відповіді.

Адміністратор:

- авторизація адміністратора;
- додавання, редагування предмету;
- додавання, редагування міста;
- видалення профілю репетитора.

Управління профілем користувача:

- створення профілю користувача;
- редагування профілю користувача.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

− статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування. Для статичного тестування використовувалась платформа Embold;

− функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

− мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з використанням наскрізного тестування (End-to-end), з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- мануальне тестування на відповідність функціональним вимогам;
- тестування веб-застосунку на всіх сучасних браузерах (Google Chrome, Edge, Mozilla Firefox, Safari);
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування на пристроях з екранами різного розміру, для перевірки адаптивності веб-застосунку;
- тестування інтерфейсу користувача;
- тестування зручності використання.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ПОШУКУ РЕПЕТИТОРІВ

Керівництво користувача

КПІ.ІТ-0221.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Анастасія СКАЧКОВА

Київ – 2024

ЗМІСТ

1 Призначення програми

«Вебзастосунок для пошуку репетиторів» - це вебзастосунок для пошуку репетиторів. Метою проєкту є підвищення ефективності пошуку та підбору розкладу занять з репетитором. Такий інструмент значно полегшить процес пошуку відповідних спеціалістів, враховуючи індивідуальні потреби та вподобання користувачів. Однією з ключових особливостей вебзастосунку є можливість організації та управління навчальним процесом. Після проведення уроку репетитори зможуть виставляти завдання учням, що сприятиме закріпленню отриманих знань та розвитку навичок. Учні матимуть змогу прикріплювати відповіді до завдань безпосередньо у застосунку, що забезпечить зворотній зв’язок та можливість коригування навчального плану у режимі реального часу.

Програмне забезпечення запускається за допомогою веб-браузера.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– Потужність процесора: рекомендовано AMD Ryzen 5 або вищі покоління;

– Наявність доступу до Інтернету;

– Обсяг оперативної пам’яті: рекомендовано 16 Гб або більше;

– Операційна система: рекомендовано використовувати Microsoft Windows, MacOS або GNU Linux.

– Браузер: рекомендовано використовувати актуальну версію Google Chrome версією 89+, Mozilla Firefox версією 87+ або Safari версією 13+.

2.2 Завантаження застосунку

Для локальної збірки та запуску застосунку, достатньо завантажити код з репозиторію проекта, зібрати та запустити проект за допомогою програмного середовища ASP.NET Core, рекомендовано в IDE VisualStudio 2017.

2.3 Перевірка коректної роботи

Після успішного запуску проекту програмного продуку, користувач може побачити головну сторінку вебзастосунку для пошуку репетиторів.

3 Виконання програми

3.1 Неавторизований користувач

При запуску програмного застосунку неавторизованому користувачу буде відображено головну сторінку, що відповідає за пошук репетиторів. На ній користувач має можливість здійснити пошук за опціями фільтрації, а саме місто, тип заняття, вартість, предмет, або знайти репетитора за частиною повного ім’я, а також переглянути детальну інформацію обраної картки репетитора. Головну сторінку неавторизованого користувача відображено на рисунку 3.1.

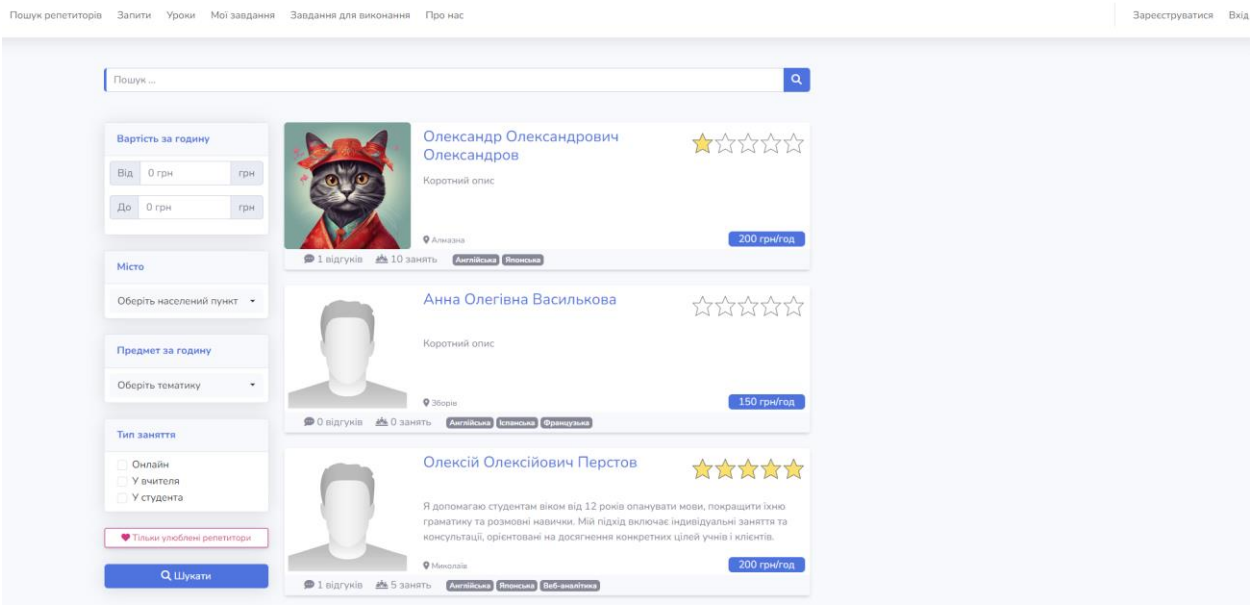


Рисунок 3.1 – Головна сторінка вебзастосунку неавторизованого користувача

Для реєстрації користувачу необхідно натиснути на кнопку «Зареєструватися» в правому верхньому кутку сторінки. Далі необхідно заповнити дані реєстрації, а саме пошту, пароль і натиснути на кнопку реєстрації. Сторінку реєстрації відображено на рисунку 3.2.

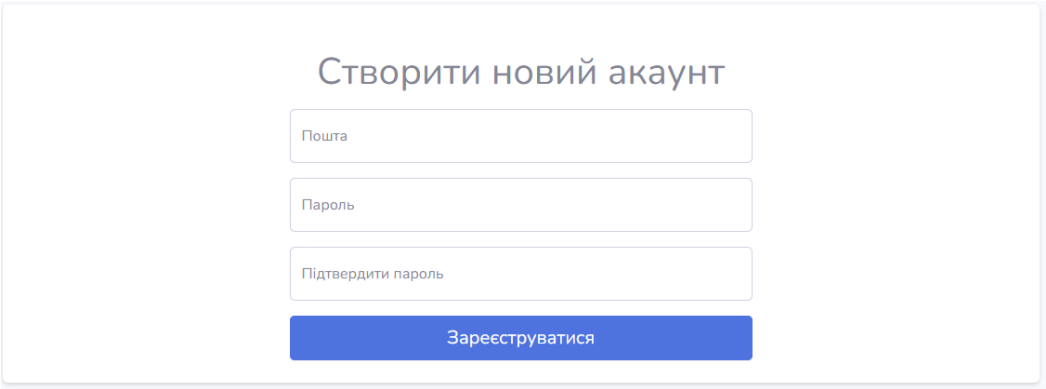


Рисунок 3.2 – Сторінка реєстрації

Для авторизації користувачу необхідно натиснути на кнопку «Вхід» в правому верхньому кутку сторінки. Далі необхідно заповнити дані реєстрації, а саме пошту, пароль і натиснути на кнопку авторизації. Сторінку авторизації відображено на рисунку 3.3.

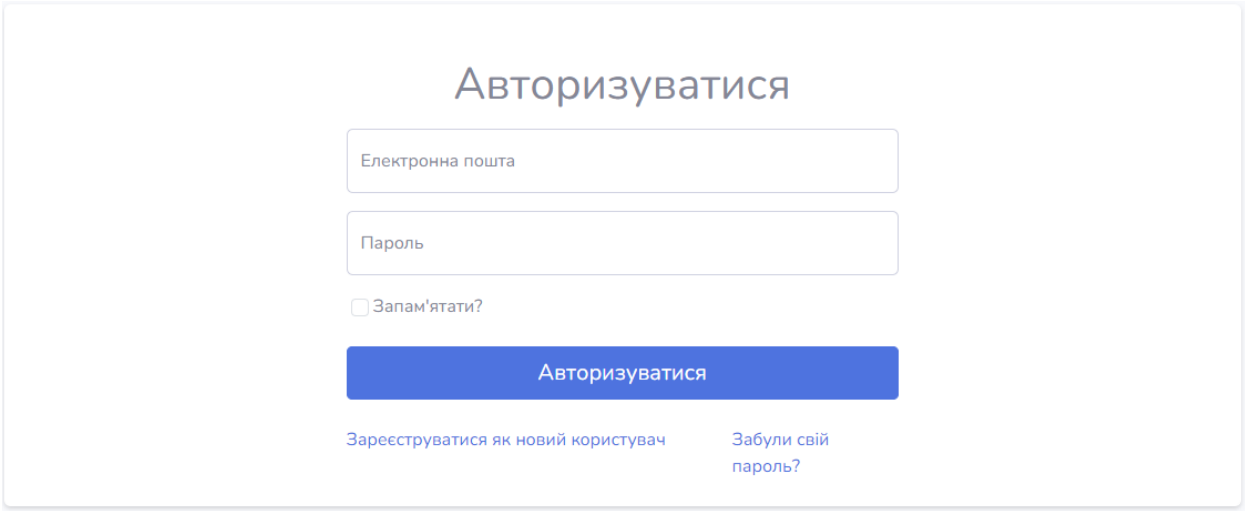


Рисунок 3.3 – Сторінка авторизації

3.2 Авторизований користувач

При запуску програмного застосунку авторизованому користувачу буде відображено головну сторінку, що відповідає за пошук репетиторів. На ній користувач має можливості здійснити пошук за опціями фільтрації, а саме місто, тип заняття, вартість, предмет, або знайти репетитора за частиною повного ім’я, переглянути список улюблених репетиторів, переглянути детальну інформацію

обраного профіля репетитора, натиснувши на нього, де можливо додати запит на заняття або додати репетитора улюблені. Головну сторінку авторизованого користувача відображено на рисунку 3.4.

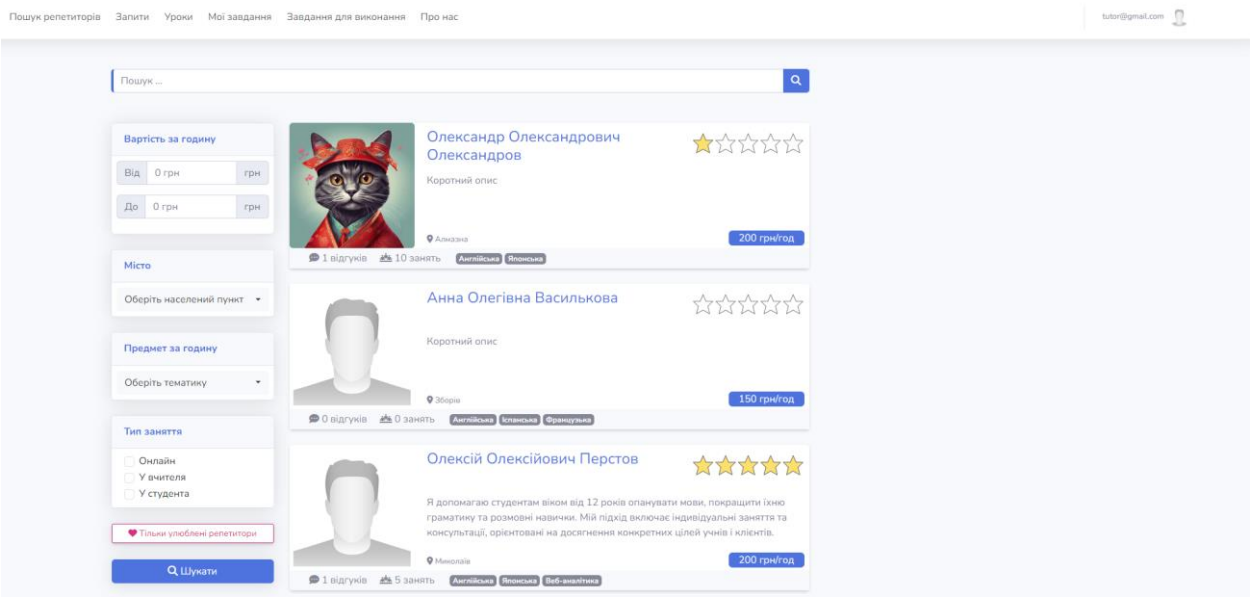


Рисунок 3.4 – Головна сторінка вебзастосунку авторизованого користувача

Для відображення повного ім’я користувача у відгуках та колонках, де відображається учень, користувач має створити профіль користувача, натиснувши на свою пошту у правому верхньому кутку та перейшовши за посиланням «Профіль». Користувач має ввести ім’я, прізвище, по батькові, обрати своє місто та завантажити свій аватар, після чого натиснути кнопку збереження. Сторінка створення профілю користувача відображена на рисунку 3.5

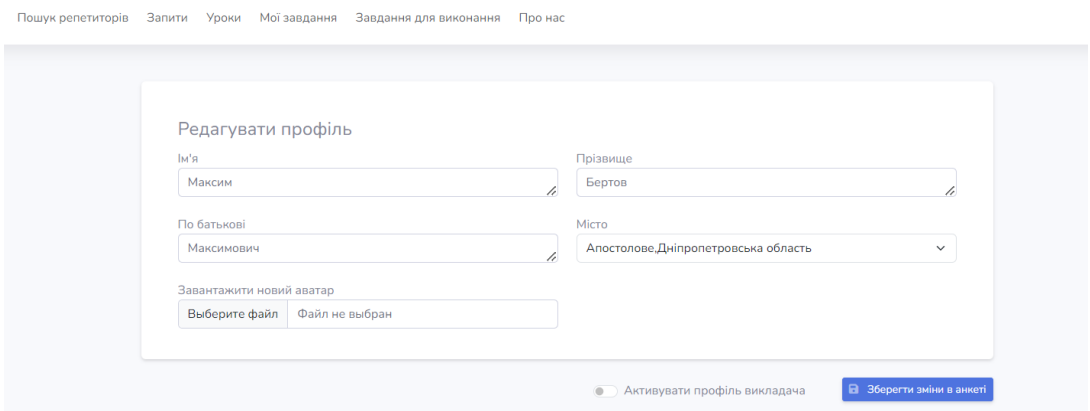


Рисунок 3.5 – Створення профілю користувача

Для створення завдань, отримання запитів на заняття від учнів користувач має створити профіль репетитора. Для цього у користувача має бути профіль користувача, користувач має перейти на сторінку профілю та активувати профіль користувача відповідним чекбоксом. Далі користувач має обрати предмети викладання, обрати тип заняття, ввести ціну за годину, домашню адресу, короткий опис, про себе та обрати години викладання в календарі. Сторінка створення профілю репетитора та календаря відображені на рисунку 3.6 – 3.7.

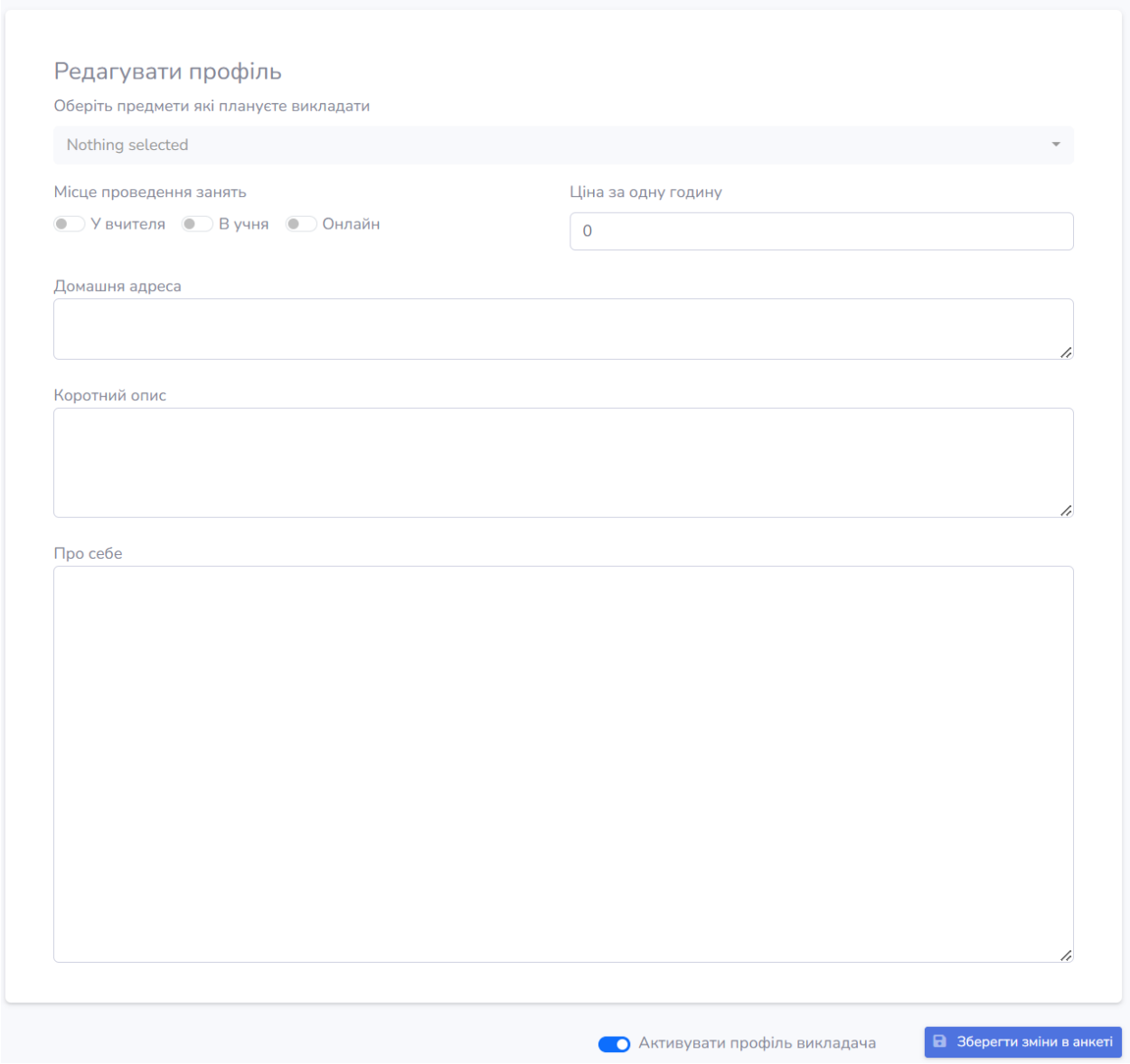


Рисунок 3.6 – Створення профілю репетитора

Рисунок 3.7 – Сторінка створення календаря

Для перегляду списку улюблених репетиторів користувач має натиснути кнопку улюблених репетиторів на головній сторінці, кнопка стане рожевою, і натиснути кнопку пошуку. Перегляд списку улюблених репетиторів відображено на рисунку 3.8.

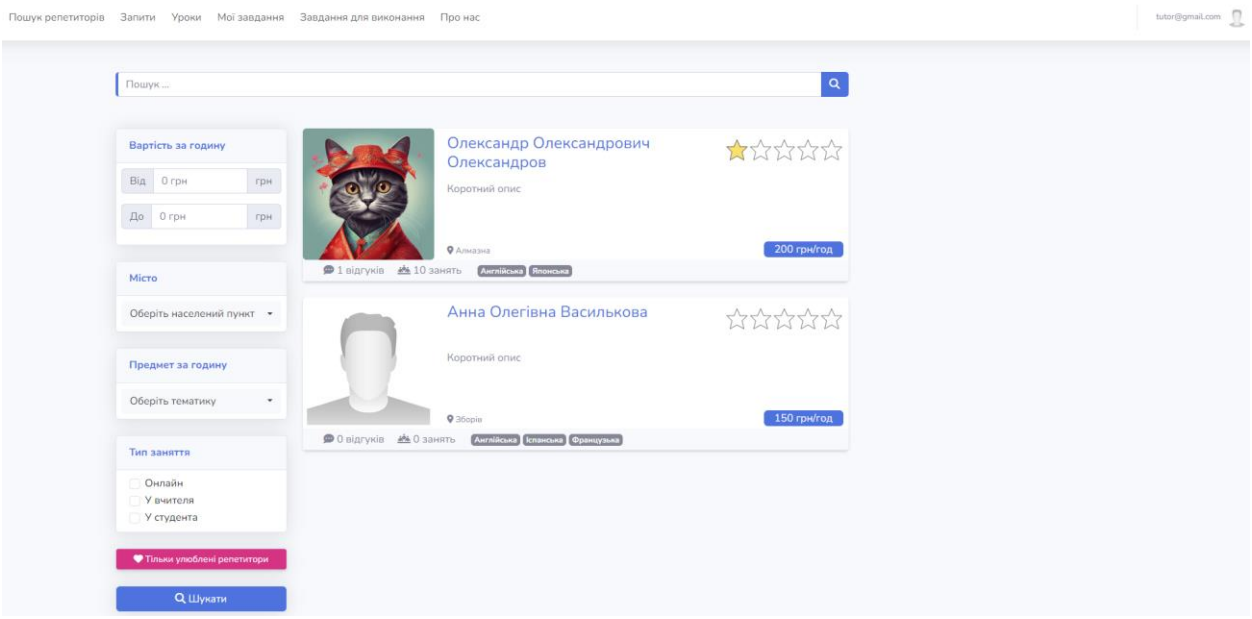


Рисунок 3.8 – Перегляд списку улюблених репетиторів

Для перегляду профілю репетитора необхідно натиснути на картку обраного репетитора. Перегляд профілю репетитора та перегляд календаря репетитора і відгуків у профілі репетитора відображено на рисунках 3.9 – 3.10.

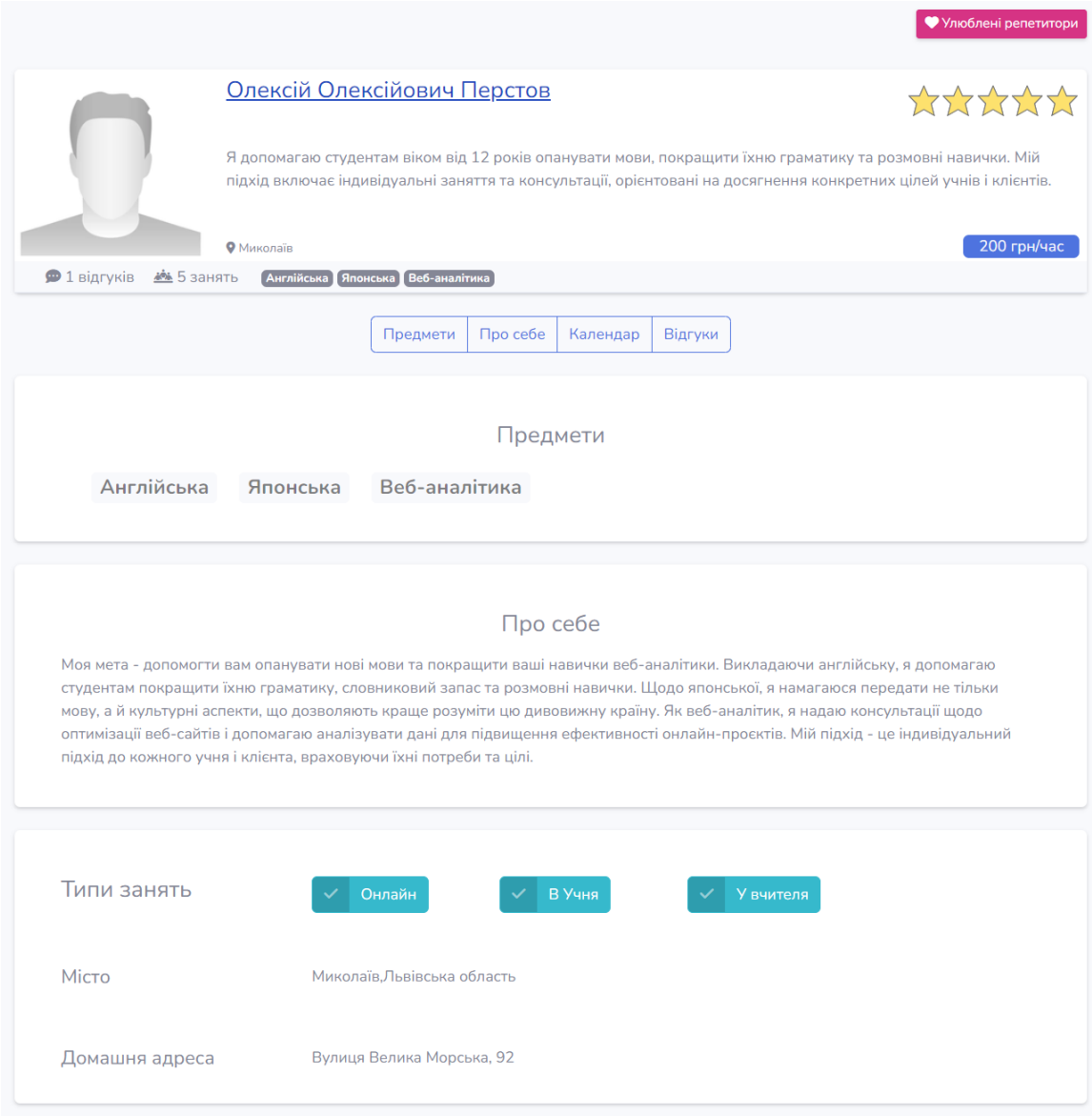


Рисунок 3.9 – Перегляд профілю репетитора

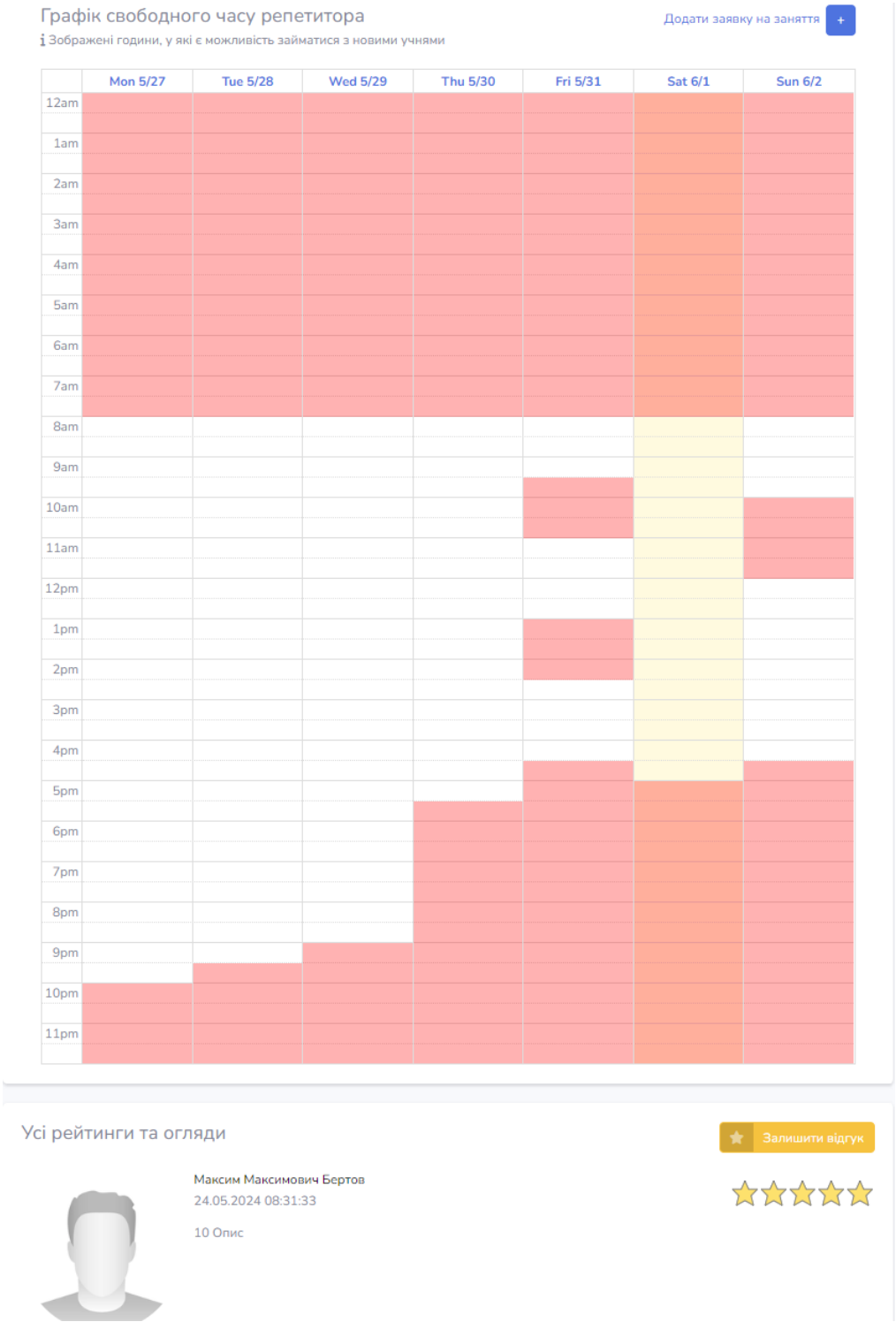


Рисунок 3.10 – Перегляд календаря репетитора і відгуків

Авторизований користувач може додати репетитора в улюблені, натиснувши кнопку улюблених репетиторів у верхньому правому кутку профіля, що змінить колір на рожевий після додавання. Додавання репетитора в улюблені відображено на рисунку 3.11.

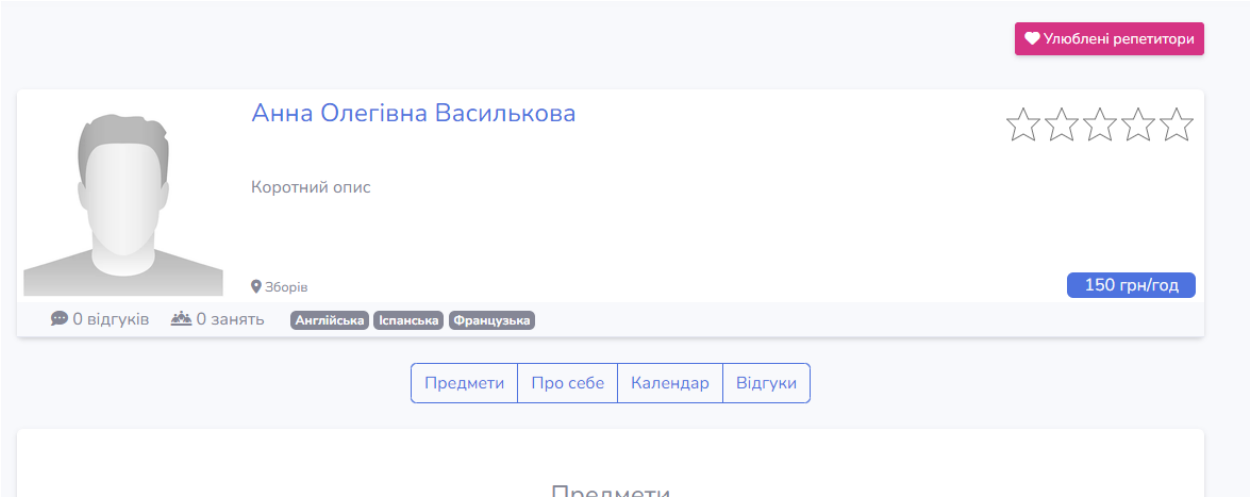


Рисунок 3.11 – Додавання репетитора в улюблені

Авторизований користувач може додати запит на заняття, обравши зручний час в календарі, або натиснувши на кнопку у правому верхньому кутку календаря. Користувачу буде відображено форму запису до репетитора, як на рисунку 3.12, з можливістю обрати предмет, час, ввести коментар до заняття і натиснути кнопку підтвердження запису. Запис до репетитора відобразиться на сторінці запитів.

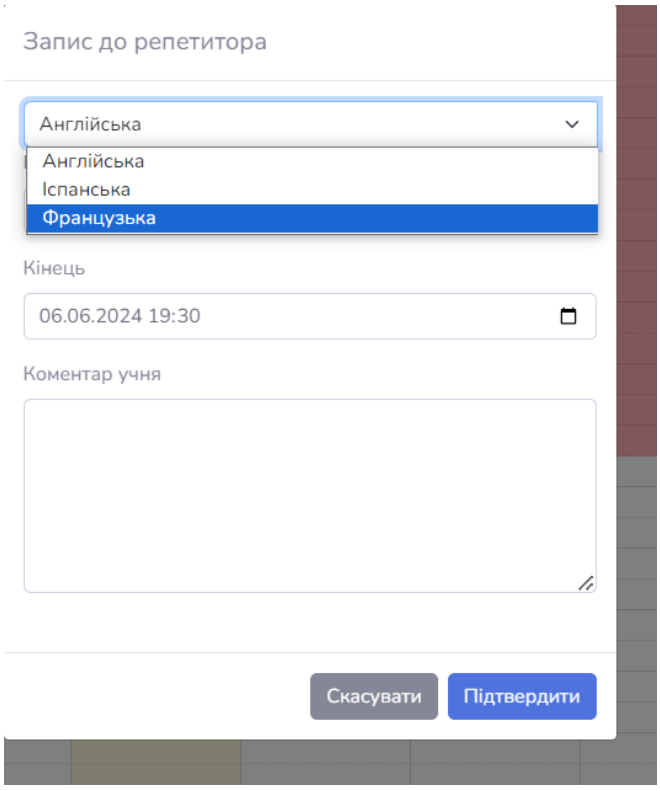


Рисунок 3.12 – Форма запису до репетитора

Користувач може додати опис репетитора та поставити рейтинг репетитору з яким у нього було хоча б одне заняття. Сторінка відгуку відображена на рисунку 3.13.

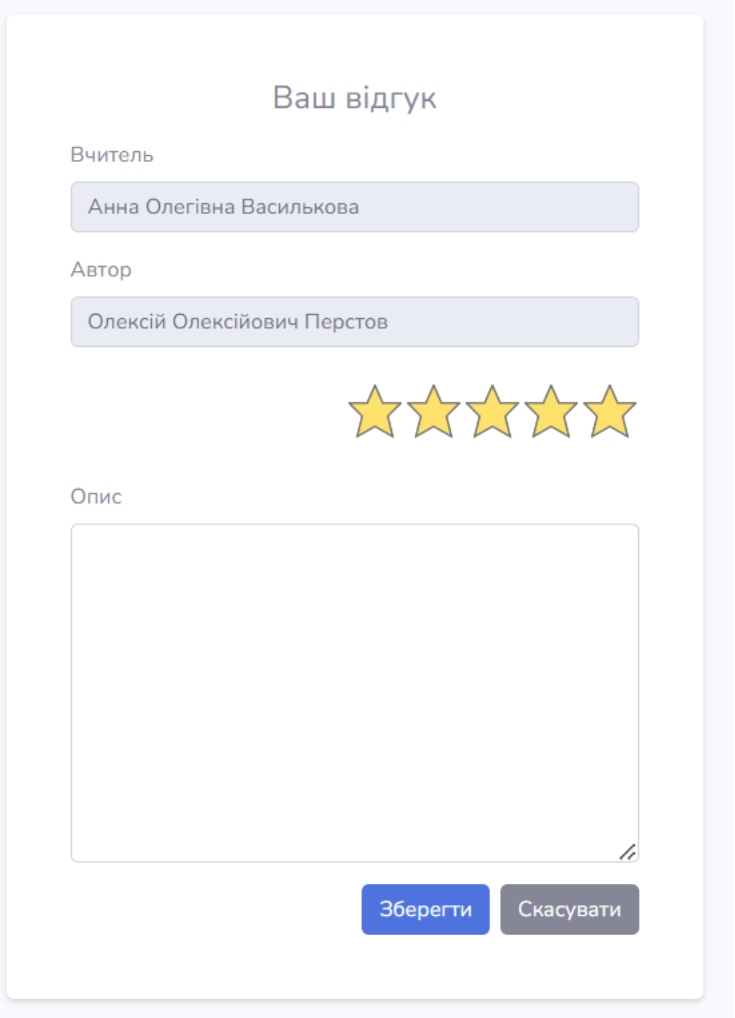


Рисунок 3.13 – Сторінка відгуку

Авторизований користувач може переглянути сторінку запитів. Для цього користувач має натиснути посилання «Запити» в навігаційному меню. Сторінку запитів відображено на рисунку 3.14.

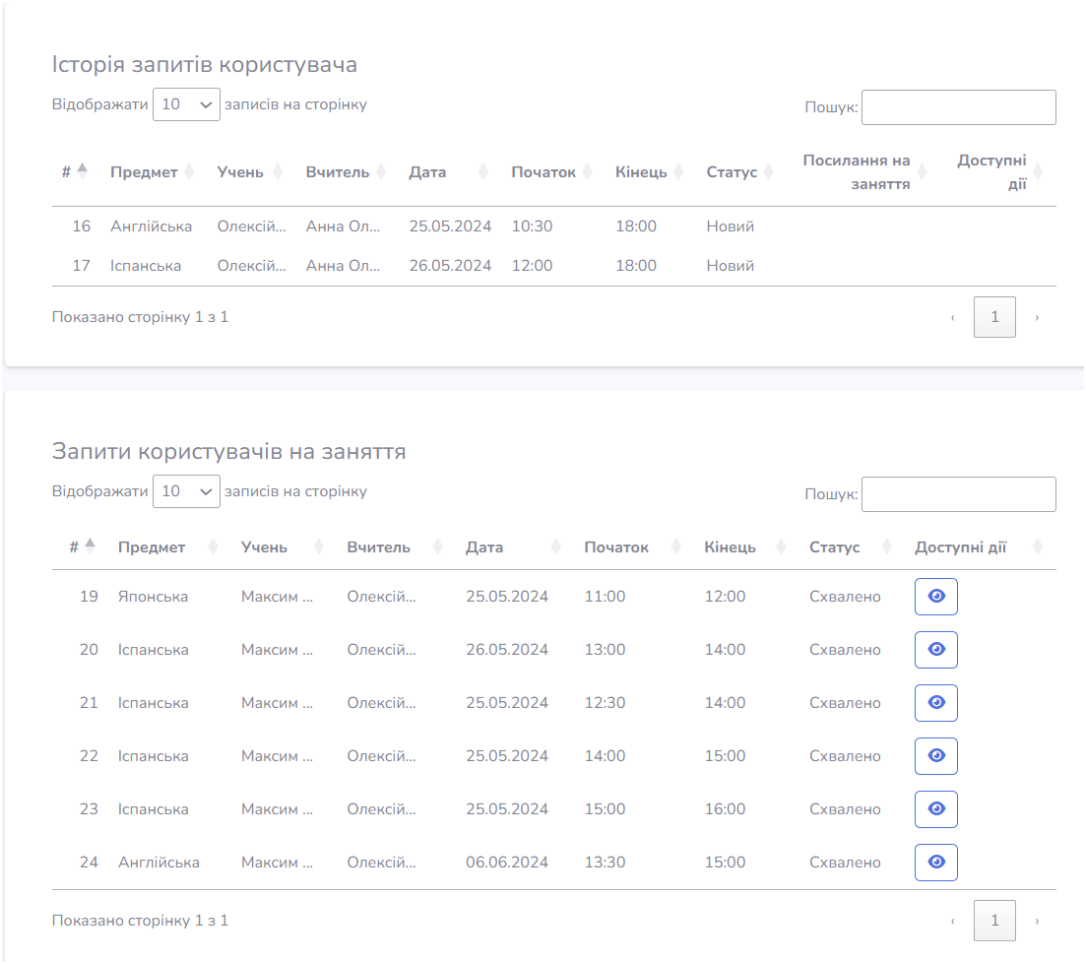


Рисунок 3.14 – Сторінка запитів

Зі сторінки запитів користувач може переглянути статус та посилання на заняття до своїх запитів або перейти на сторінку відповіді на запити до своїх занять натиснувши відповідну кнопку. На сторінці відповіді на запит користувач має надати посилання на заняття та змінити статус заняття. Сторінка відповіді на запит відображена на рисунку 3.15.

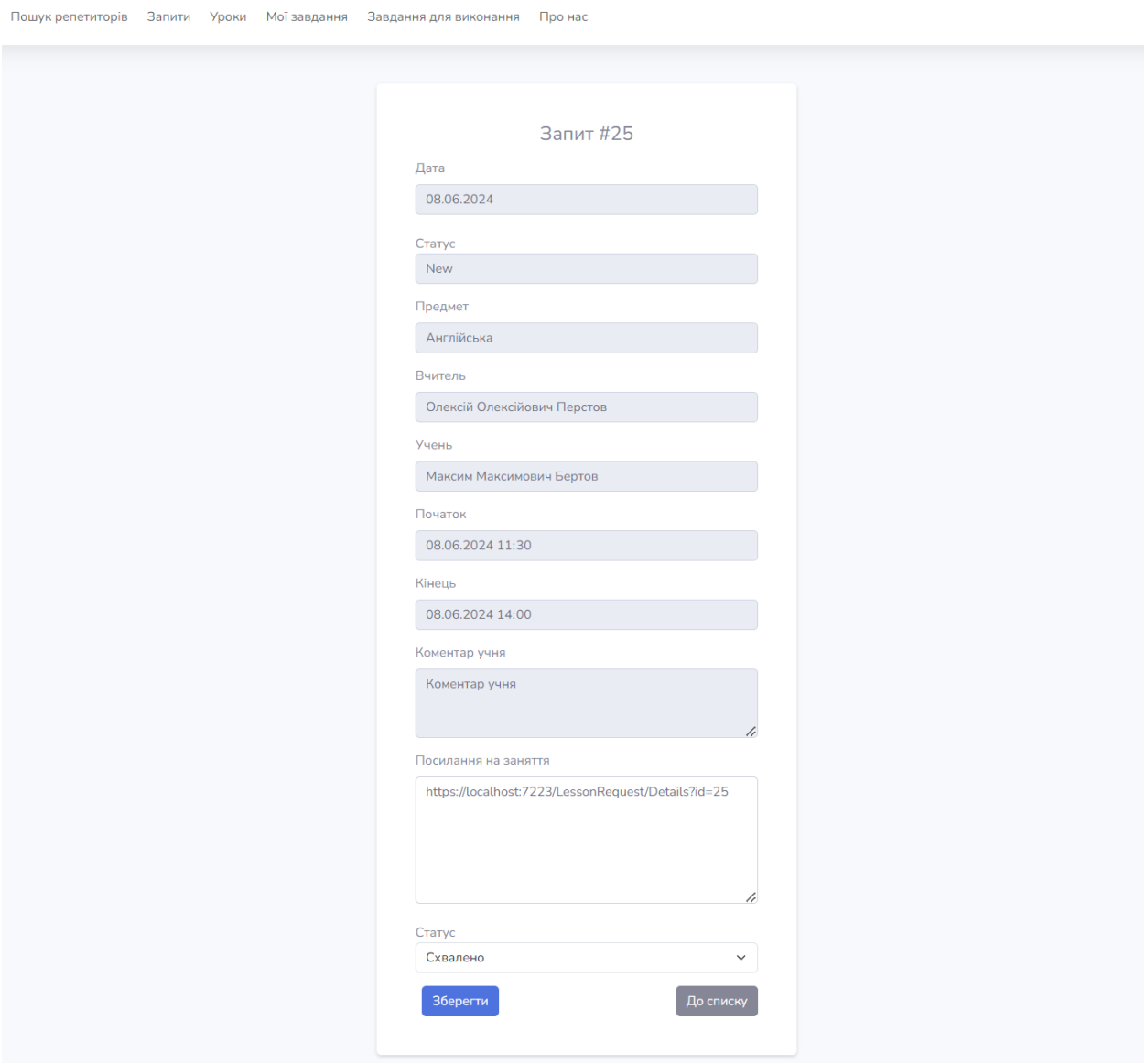


Рисунок 3.15 – Сторінка відповіді на запит

Авторизований користувач, що має профіль репетитора, може додати, редагувати або видалити завдання для учня. Для цього користувач має натиснути посилання «Мої завдання» в навігаційному меню. Сторінку моїх завдань відображено на рисунку 3.16.

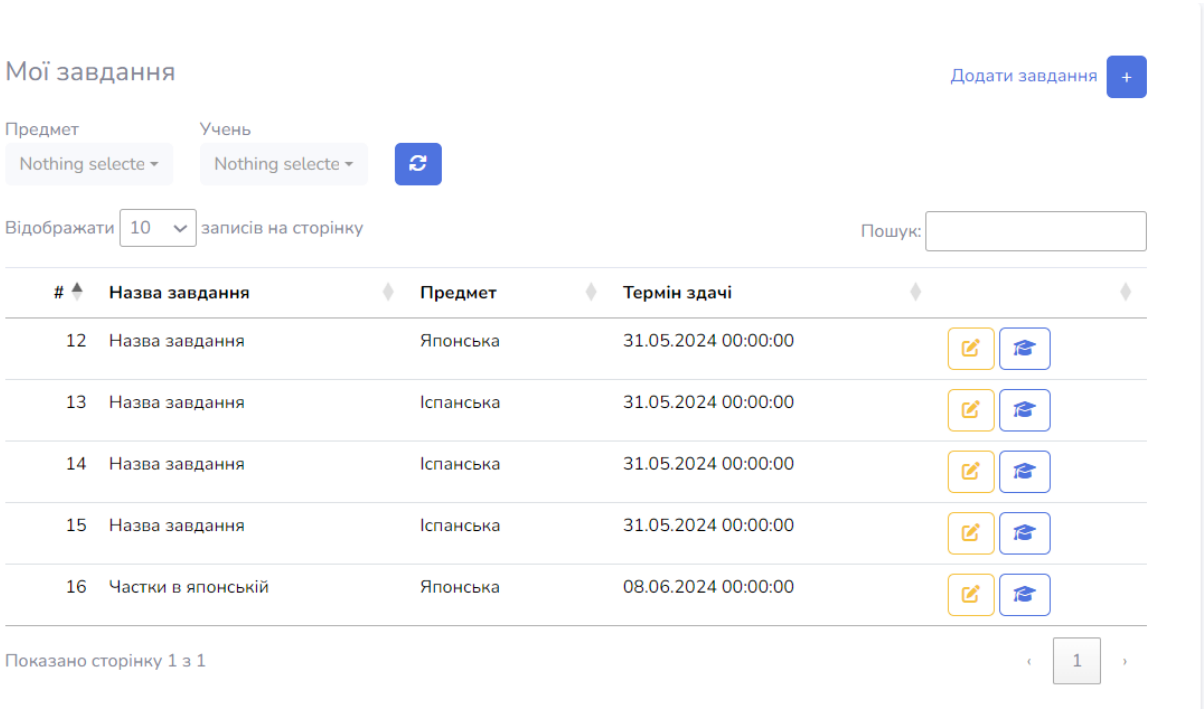


Рисунок 3.16 – Сторінка моїх завдань

Для додавання заняття учню репетитора має натиснути кнопку додавання завдання у правому верхньому кутку. Користувач має ввести назву завдання, обрати предмет та учня, ввести опис завдання. Сторінка створення завдання відображена на рисунку 3.17.

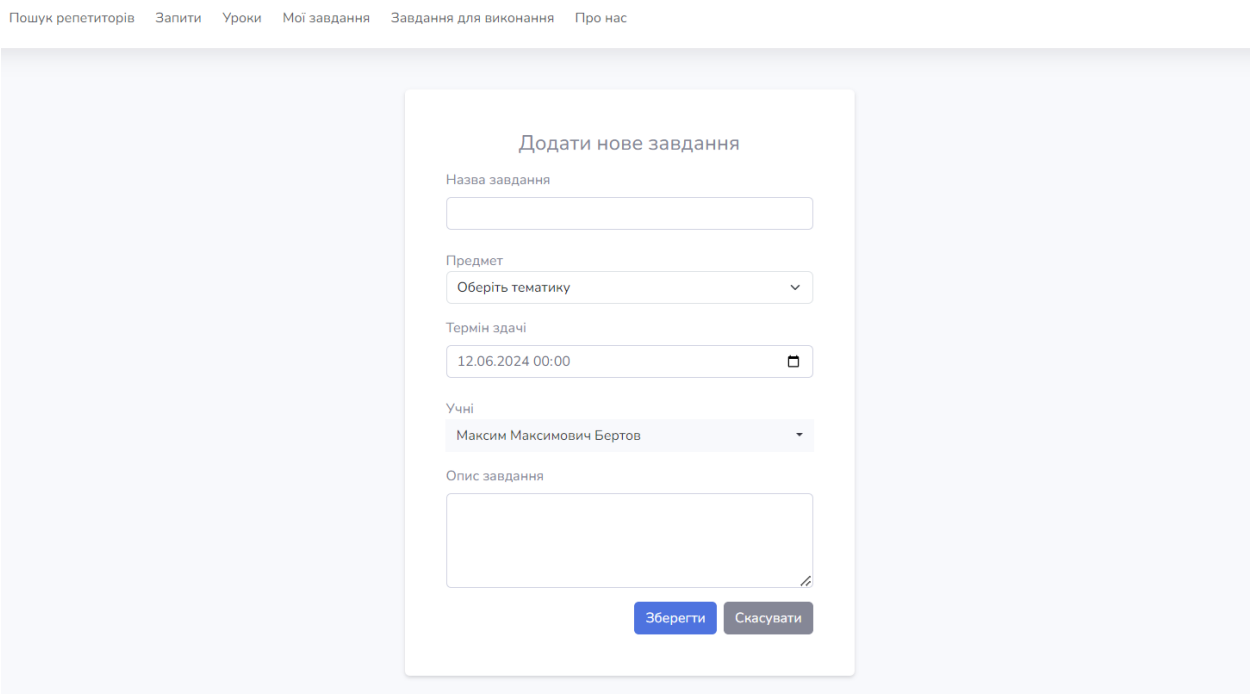


Рисунок 3.17 – Сторінка створення завдання

Після створення завдання користувач може прикріпити файл до завдання, що зображено на рисунку 3.18.

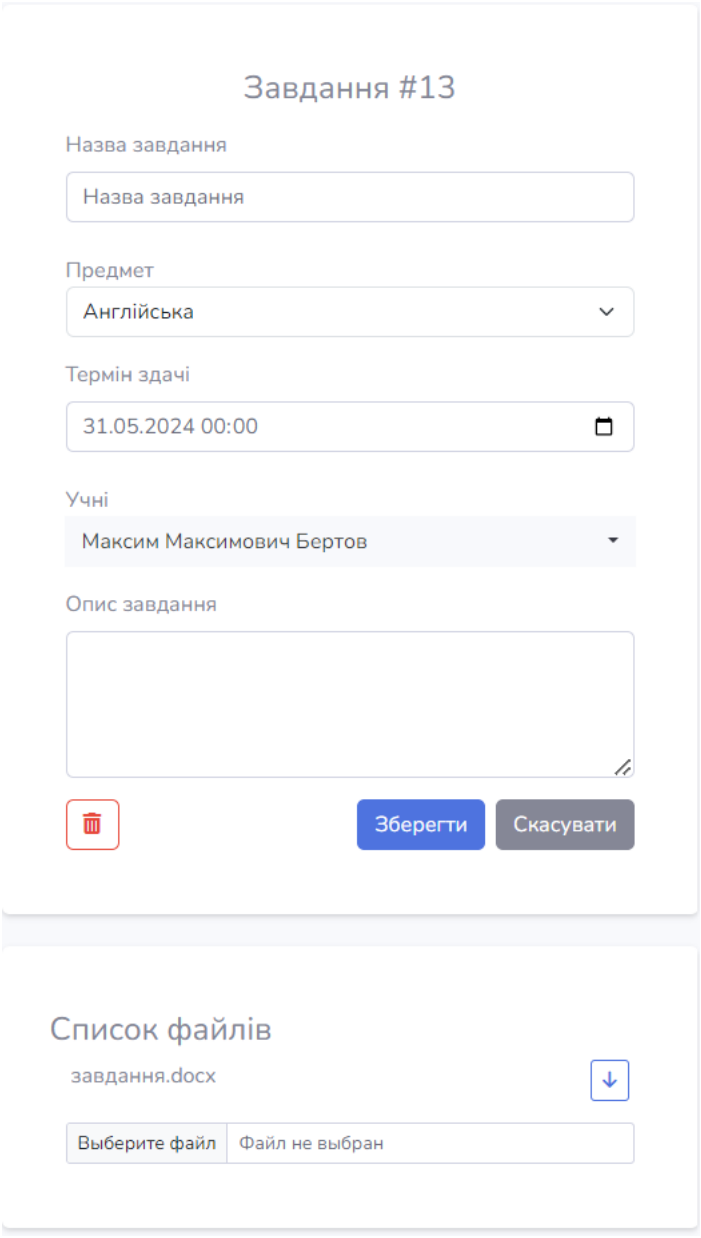


Рисунок 3.18 – Прикріплення файлу до завдання

Для видалення завдання користувач має натиснути на кнопку видалення зліва від опису завдання на сторінці завдання.

Для редагування завдання користувач має натиснути кнопку редагування з правої сторони таблиці. Після зміни даних завдання користувач має натиснути

кнопку збереження. Кнопка скасування дозволить перейти на сторінку своїх завдань. Сторінка редагування завдання відображена на рисунку 3.19.

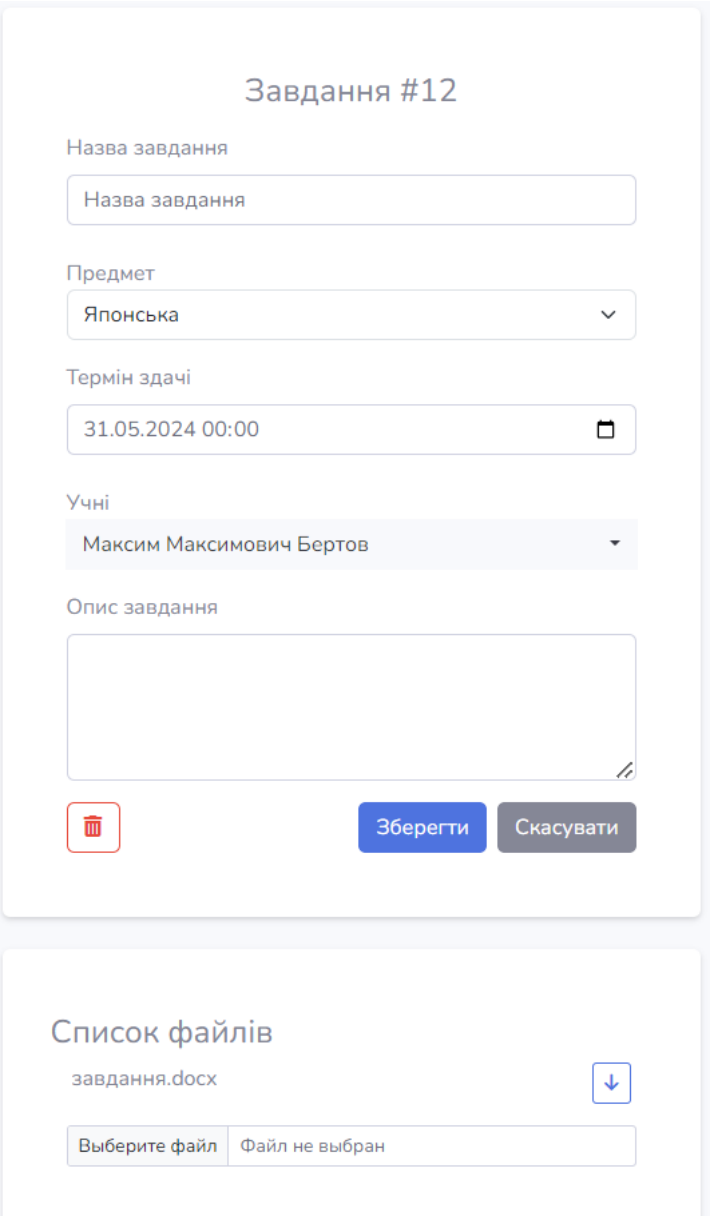


Рисунок 3.19 – Сторінка редагування завдання

Також поряд з правої сторони є кнопка перегляду статусу обраного завдання та відповіді, якщо вона вже надана учнем. Сторінка перегляду статусу обраного завдання та відповіді учня відображена на рисунку 3.20.

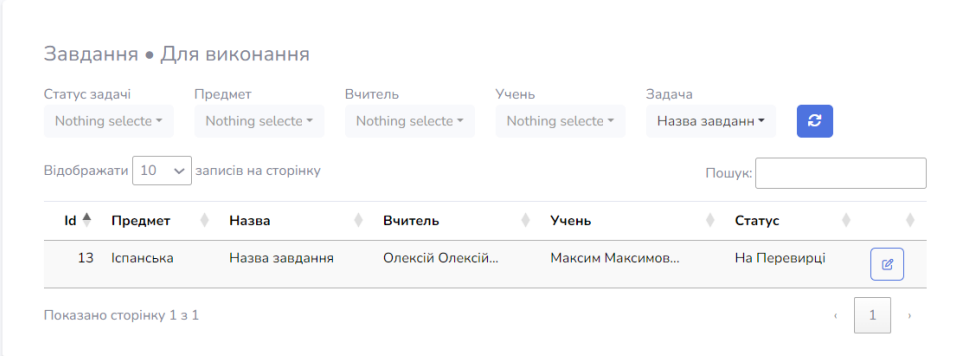


Рисунок 3.20 – Сторінка перегляду відповіді обраного учня

Користувач може перевірити учня, натиснувши на кнопку перевірки з правої сторони таблиці. Користувач, що є репетитором може ввести коментар, змінити статус виконання завдання та додати файли за потреби. Після заповнення даних користувач має натиснути кнопку збереження. Кнопка скасувати дозволяє перейти на сторінку виставлених завдань. Сторінка перевірки відповіді учня відображена на рисунку 3.21.

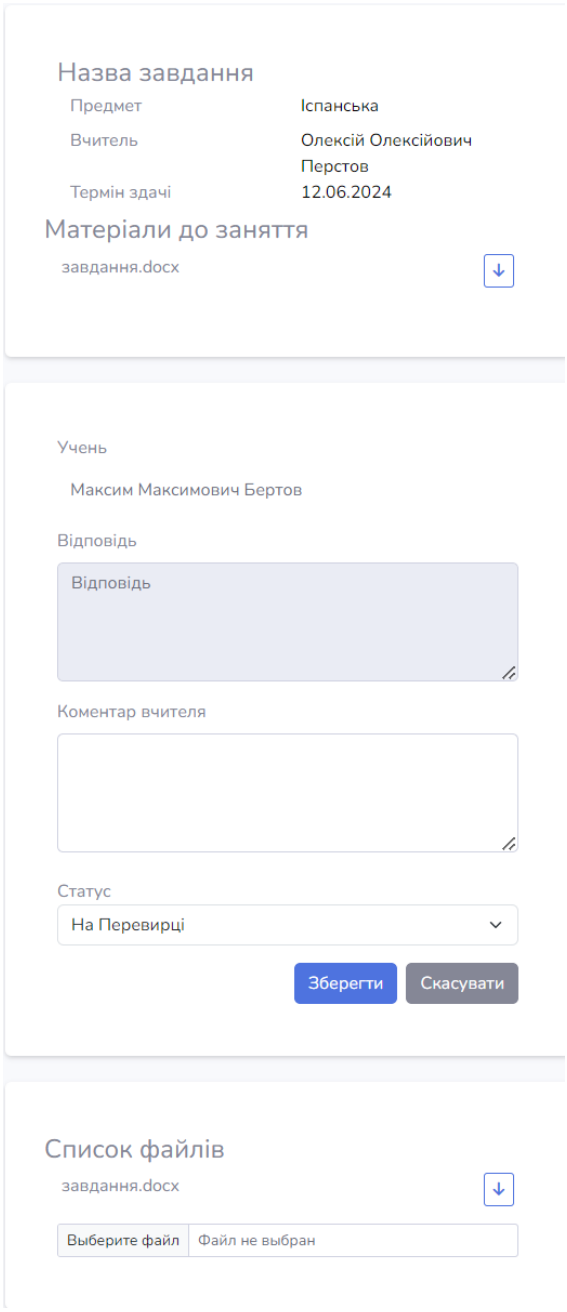


Рисунок 3.21 – Сторінка перевірки відповіді учня

Авторизований користувач може перейти на сторінку виставлених завдань, натиснувши посилання «Завдання для виконання» в навігаційному меню. Сторінка дозволяє переглянути статус виконання кожного завдання, перевірити відповідь на свої завдання або створити відповідь на завдання від репетиторів, натиснувши кнопку з правої сторони таблиці. Сторінку виставлених завдань відображено на рисунку 3.22.

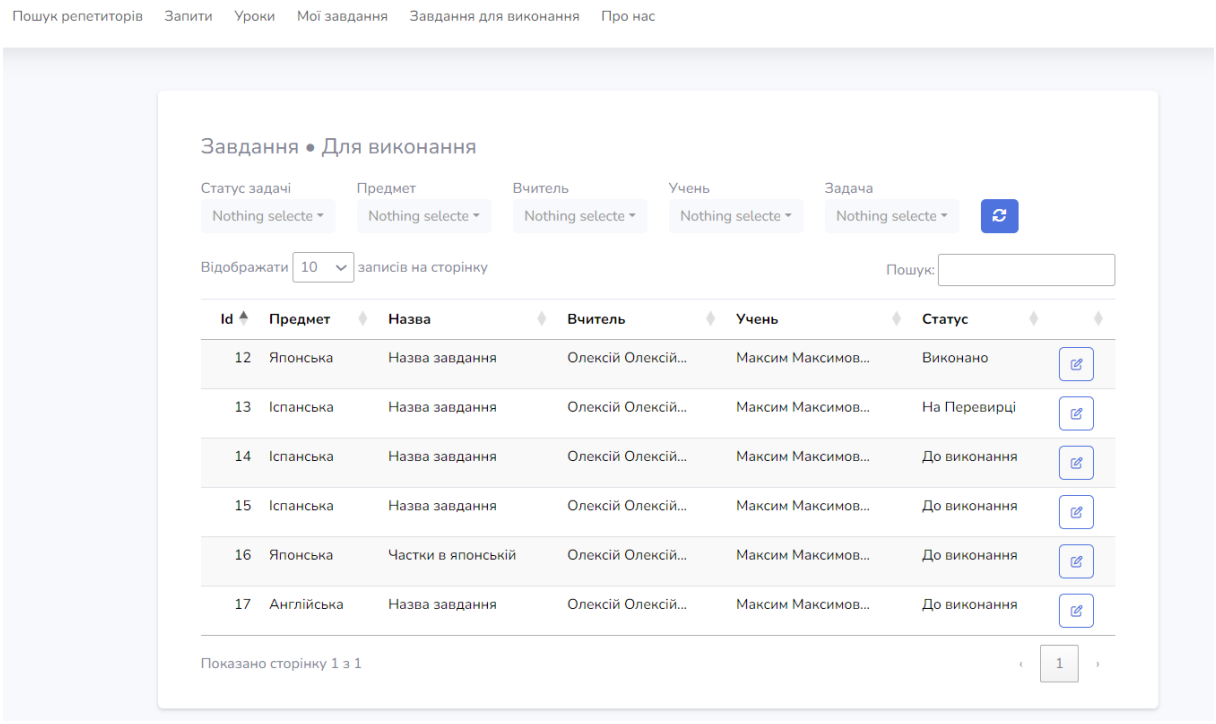


Рисунок 3.22 – Сторінка виставлених завдань

Під час створення відповіді на завдання користувач може ввести дані відповіді та прикріпити файл. Після цього необхідно натиснути кнопку збереження відповіді. Сторінка створення відповіді відображена на рисунку 3.23.

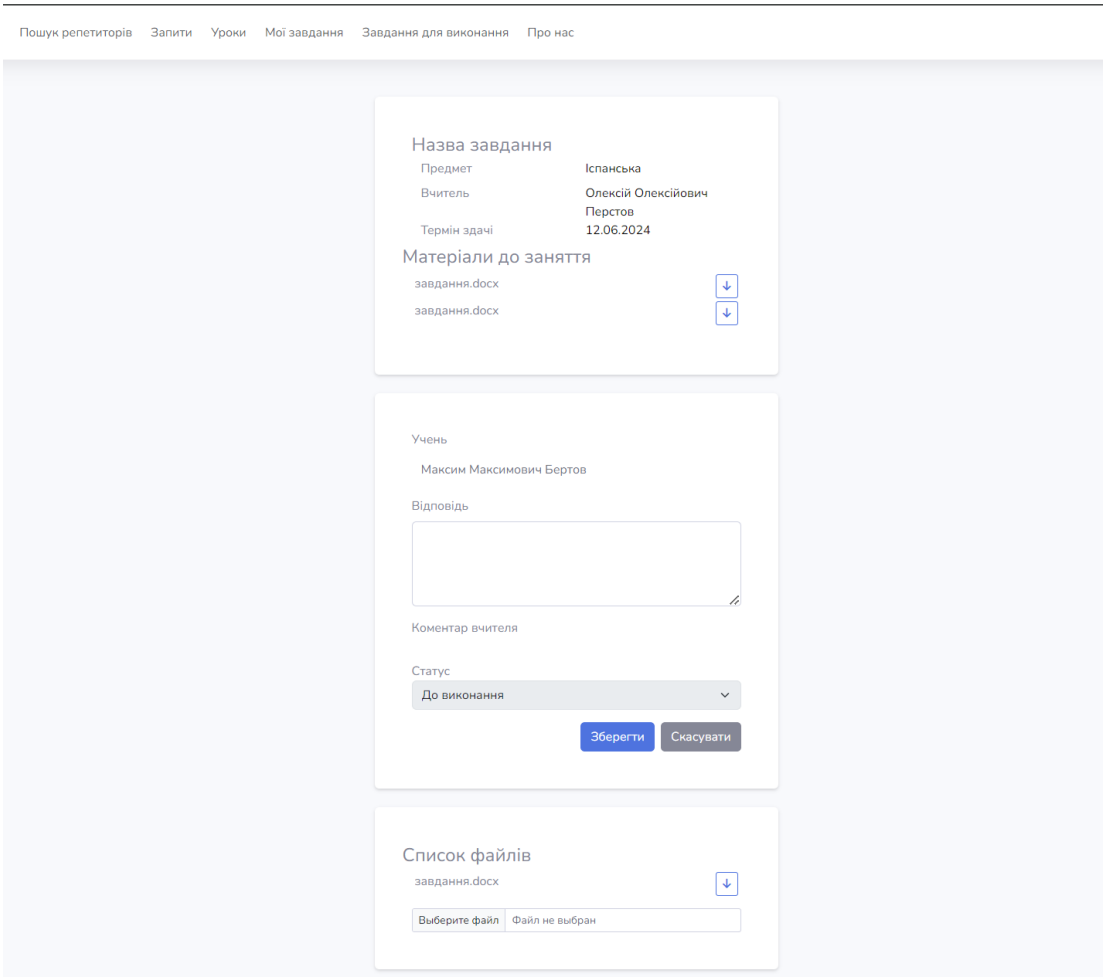


Рисунок 3.23 – Сторінка створення відповіді

3.3Адміністратор

Адміністратор може додавати, редагувати та видаляти предмети викладання. Для цього адміністратор має натиснути у верхньому правому кутку навігаційного меню на свою пошту та обрати із випадаючого списку посилання «Список предметів». Сторінку списку предметів відображено на рисунку 3.24.

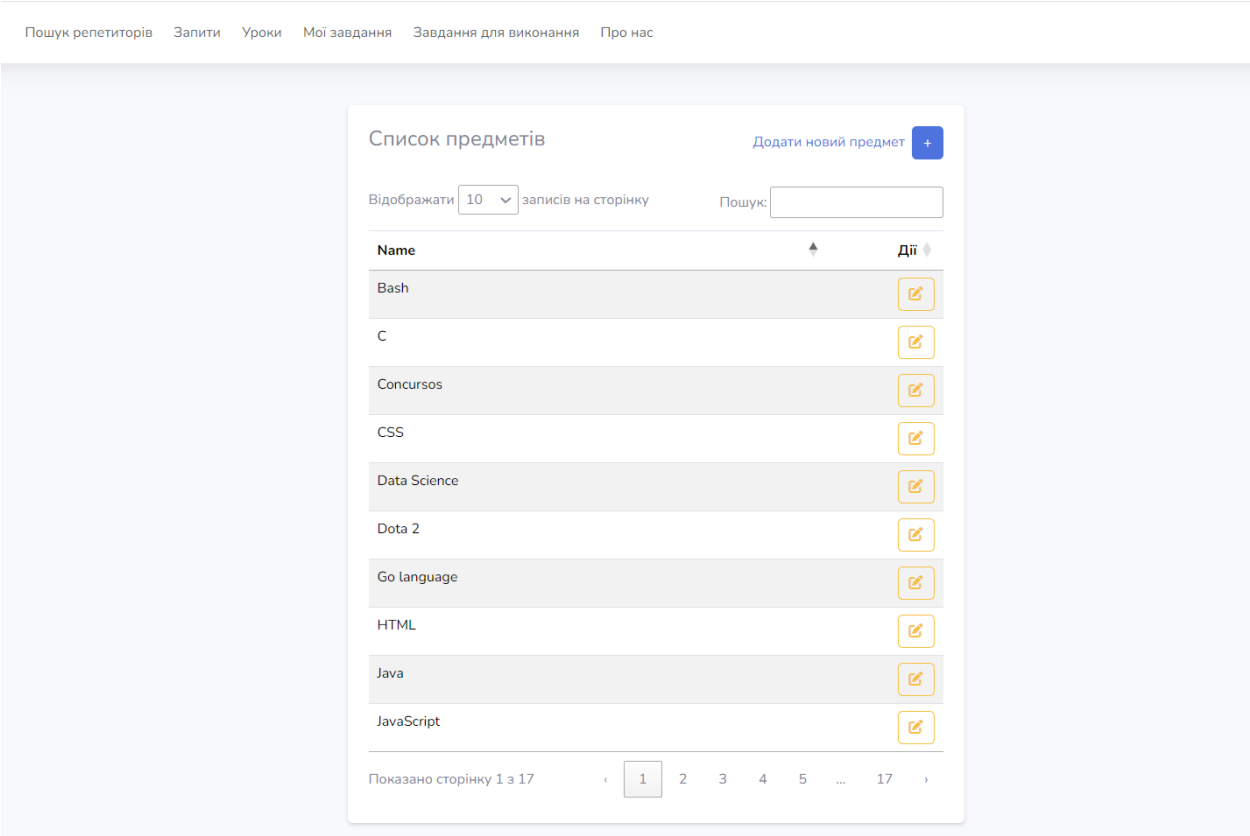


Рисунок 3.24 – Сторінка списку предметів

Для додавання предмету адміністратор має натиснути кнопку додавання предмету та ввести його назву. Після цього необхідно натиснути кнопку збереження. При натисканні кнопки скасування адміністратор повернеться на сторінку списку предметів. Сторінку додавання предмету відображено на рисунку 3.25.

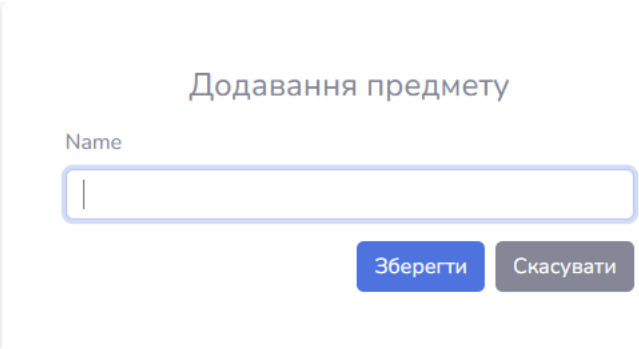


Рисунок 3.25 – Сторінка додавання предмету

Для редагування предмету необхідно натиснути навпроти ім’я предмету та ввести у поле нове ім’я предмету. Для видалення необхідно настинути кнопку видалення під полем ім’я. Сторінку редагування та видалення предмету відображено на рисунку 3.26.

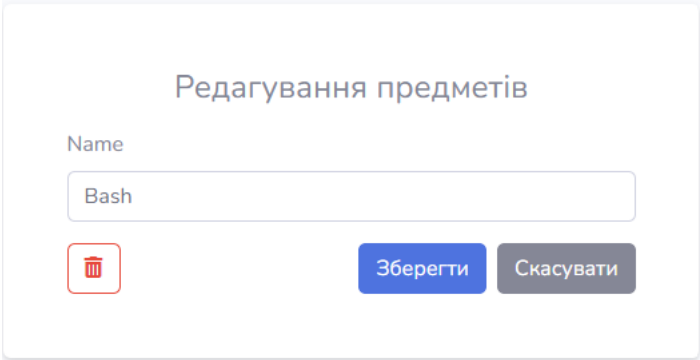


Рисунок 3.26 – Сторінка редагування та видалення предмету

За аналогією відбувається додавання , редагування та видалення міст. Відображення необхідних сторінок можна побачити на рисунках 3.27-3.29

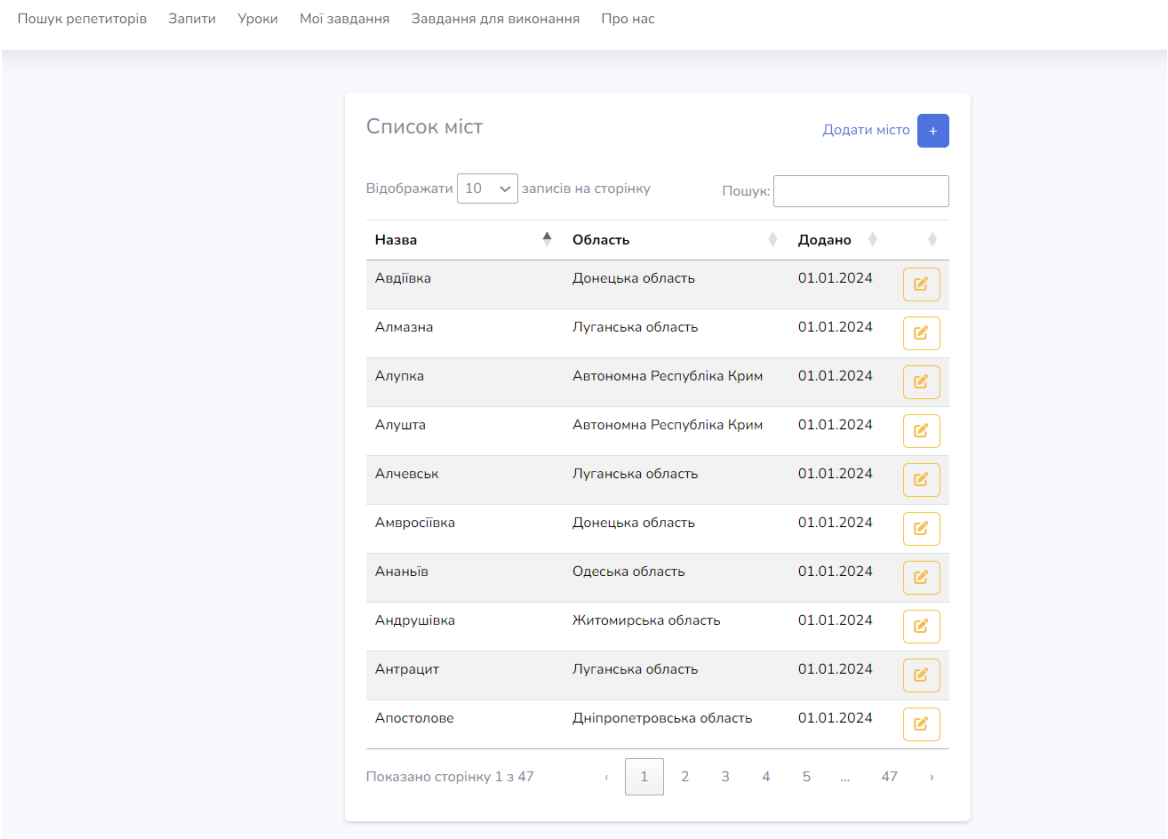


Рисунок 3.27 – Сторінка списку міст

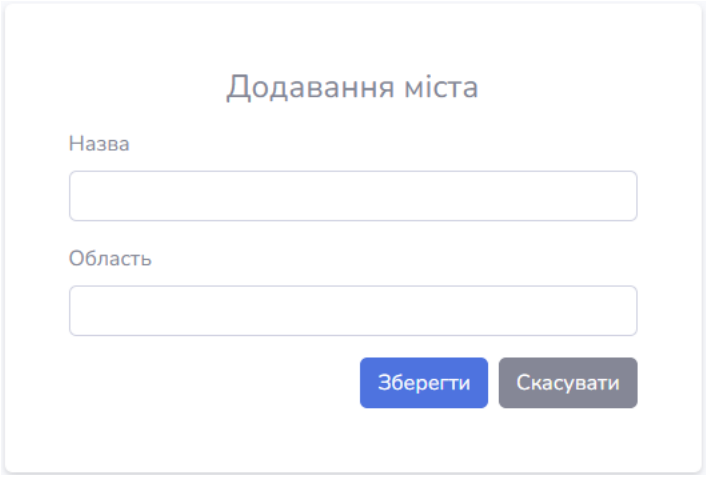


Рисунок 3.28 – Сторінка додавання міста

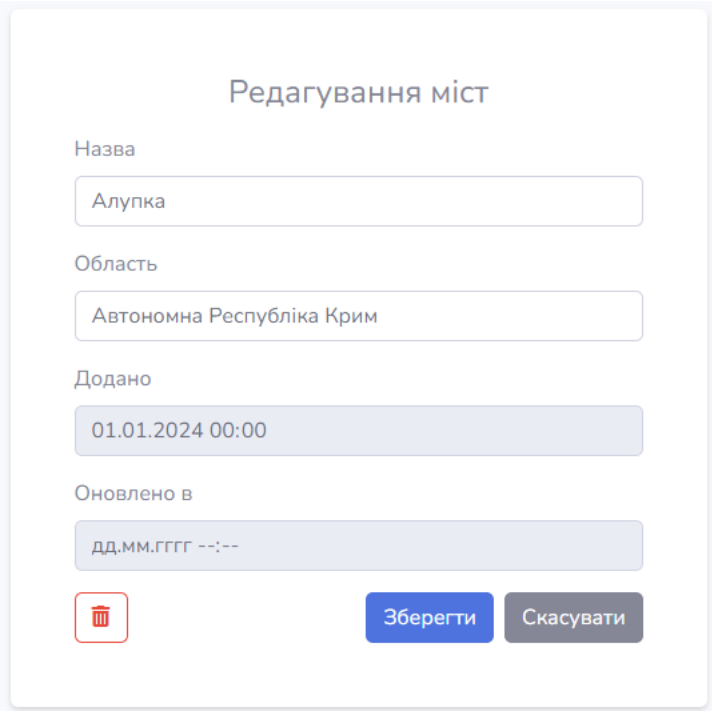


Рисунок 3.29 – Сторінка редагування та видалення міста

Для видалення профілю користувача адміністратор має перейти на сторінку видалення користувача, натиснувши в правому верхньому кутку навігаційного меню на свою пошту та обравши посилання «Видалення користувачів». Для видалення адміністратор має обрати із випадаючого списку повне ім’я необхідного кристувача та натиснути кнопку видалення. Сторінку видалення користувачів відображено на рисунку 3.30.

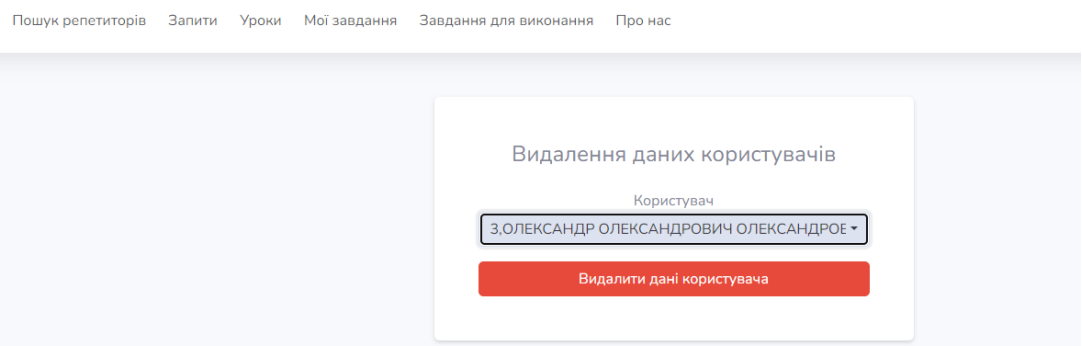


Рисунок 3.30 – Сторінка видалення користувачів

3.4 Видалений користувач

При авторизації видалений користувач буде повідомлений про видалення із вебзастосунку та матиме можливість видалити свої облікові дані, натиснувши кнопку видалення облікових даних.

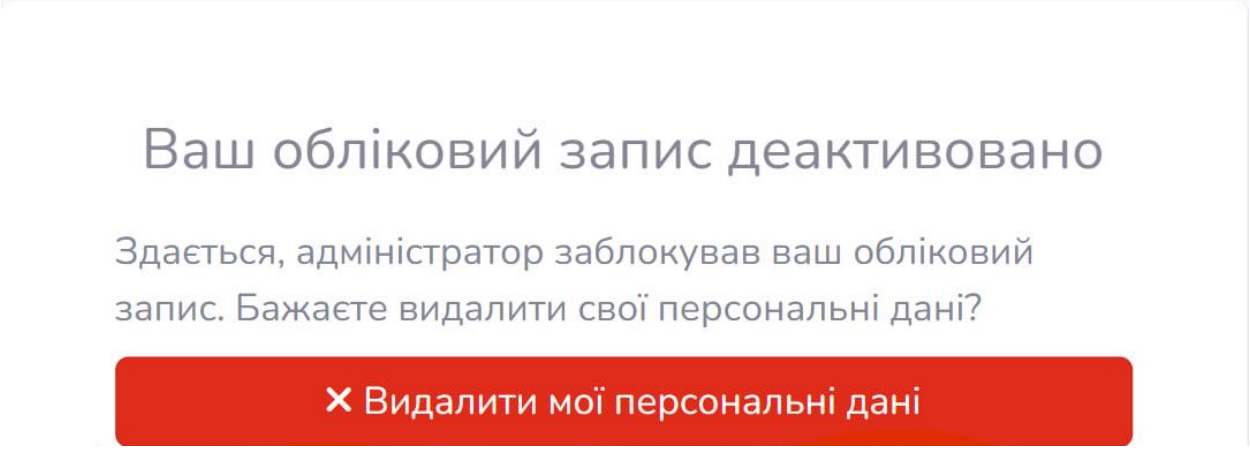


Рисунок 3.31 – Сторінка видаленого користувача

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ПОШУКУ РЕПЕТИТОРІВ

Графічний матеріал

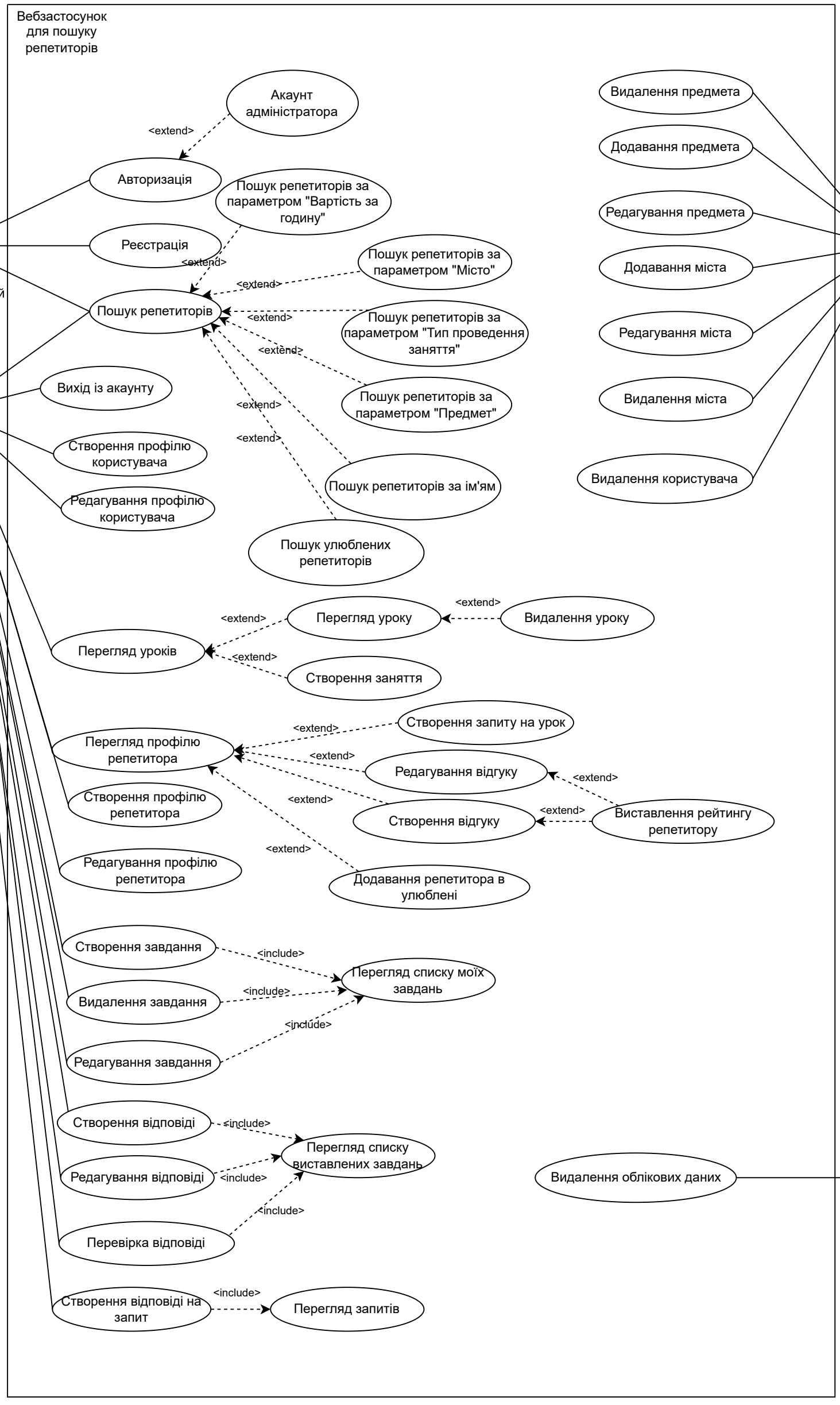
КПІ.ІТ-0221.045440.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:
_____ Катерина ЛІЩУК

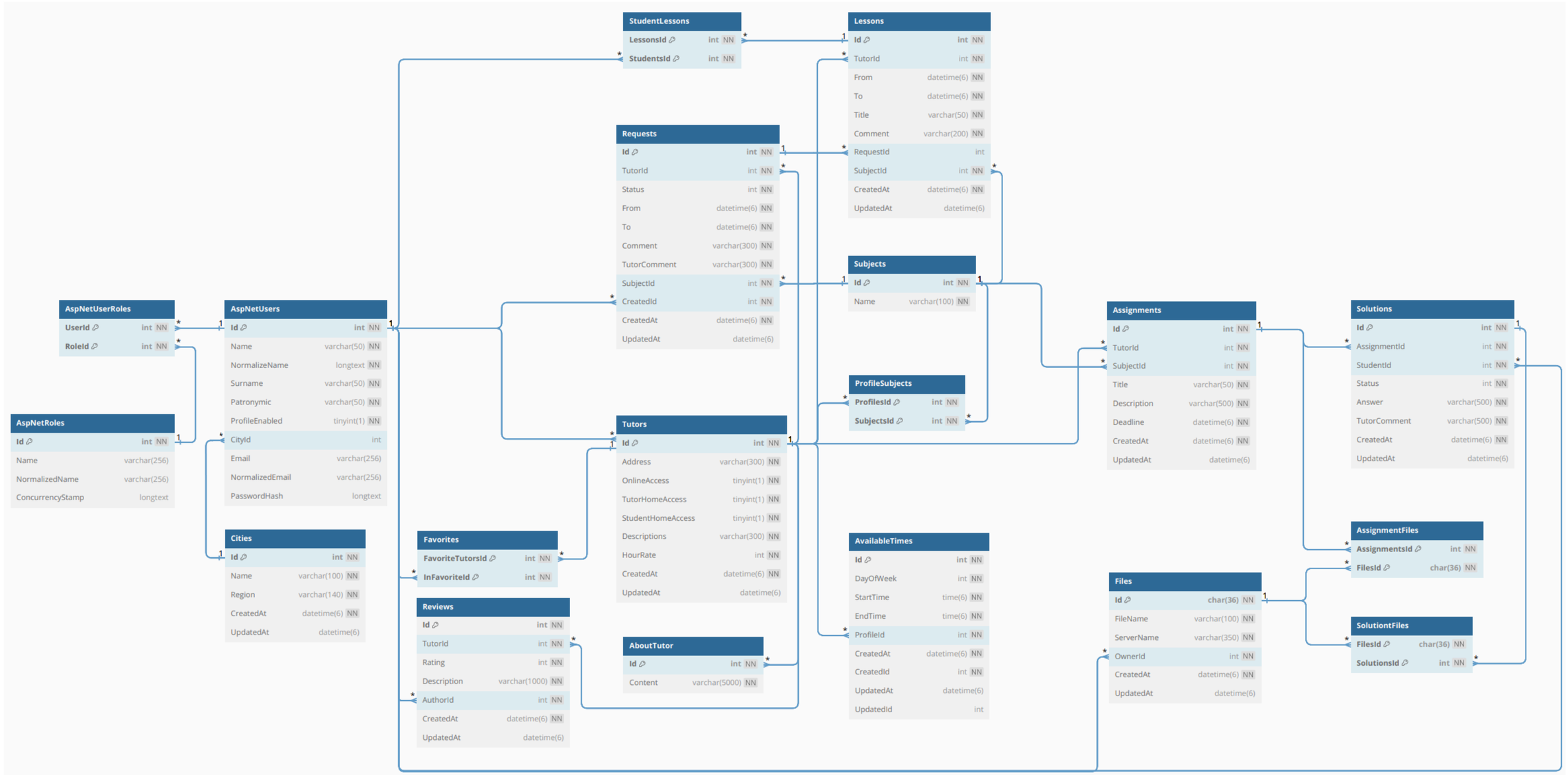
Виконавець:
_____ Анастасія СКАЧКОВА

Київ – 2024



					КПІ.ІТ-0221.045440.06.99.ССВ			
					Схема структурна варіантів використань	Лит.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підп.	Дата				
Розробив		Скачкова А.Д.				Аркуш	Аркушів	
Перевірив		Лісовиченко О.І.			Вебзастосунок для пошуку репетиторів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-02		
Т. контр.								
Н. контр.		Ліщук К.І.						
Затвердив		Жаріков Е.В.						

КПІ.ІТ-0221.045440.06.99.СБД



					КПІ.ІТ-0221.045440.06.99.СБД
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних
Розробив		Скачкова А.Д.			
Перевірив		Лісовиченко О.І.			
Т. контр.					
Н. контр.		Ліщук К.І.			Вебзастосунок для пошуку репетиторів
Затвердив		Жаріков Е.В.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-02

Літера	Маса	Масштаб

Аркуш	Аркушів

Завдання • Для виконання

Статус завдані: Nothing selecte | Предмет: Nothing selecte | Вчитель: Nothing selecte | Учень: Nothing selecte | Завдяна: Nothing selecte

Відображати 10 записів на сторінку

Пошук:

id	Предмет	Назва	Вчитель	Учень	Статус
12	Японська	Назва завдання	Олексій Олексі...	Максим Максимов...	Виконано
13	Іспанська	Назва завдання	Олексій Олексі...	Максим Максимов...	На Перевірці
14	Іспанська	Назва завдання	Олексій Олексі...	Максим Максимов...	До виконання

Уроки

Відображати 10 записів на сторінку

Пошук:

Предмет	Вчитель	Учні	Дата	Початок	Кінець	Дії
Англійська	Олексій Олексійович Перстов	Максим Максимов...	06.06.2024	13:30	15:00	
Японська	Олексій Олексійович Перстов	Максим Максимов...	25.05.2024	11:00	12:00	
Іспанська	Олексій Олексійович Перстов	Максим Максимов...	26.05.2024	13:00	14:00	
Іспанська	Олексій Олексійович Перстов	Максим Максимов...	25.05.2024	12:30	14:00	

Олексій Олексійович Перстов
Я допомагаю студентам вийти від 12 років опанувати мови, покращити їхню граматику та розмовні навички. Мій підхід включає індивідуальні заняття та консультації, орієнтовані на досягнення конкретних цілей учнів і клієнтів.
Японська Англійська Іспанська Веб-аналітика
5 занять 1 відгуків
200 грн/час
Анна Олегівна Васильєва
Коротший опис
Зобрів
150 грн/час
0 занять 0 відгуків
Англійська Іспанська Німецька Французька
Коротший опис
Автозапчастини
200 грн/час
10 занять 1 відгуків
Англійська Японська
Олексій Олексійович Перстов
Коротший опис
Автозапчастини
200 грн/час
5 занять 1 відгуків
Англійська Японська Веб-аналітика

Пошук ...

Вартість за годину

Від 0 грн грн

До 0 грн грн

Місто

Оберіть населений пункт

Предмет за годину

Оберіть тематику

Тип заняття

Онлайн
У вчителя
У студента

Тільки умовлені репетитори

Шукати

Пошук репетиторів | Заняття | Уроки | Мої завдання | Завдання для виконання | Про нас

Мої завдання

Додати завдання +

Предмет | Учень | Оцінка | Nothing selecte | Nothing selecte

Відображати 10 записів на сторінку

Пошук:

#	Назва завдання	Предмет	Термін здачі
12	Назва завдання	Японська	31.05.2024 00:00:00
13	Назва завдання	Іспанська	31.05.2024 00:00:00
14	Назва завдання	Іспанська	31.05.2024 00:00:00
15	Назва завдання	Іспанська	31.05.2024 00:00:00
16	Частки в японській	Японська	08.06.2024 00:00:00

Про себе

Моя мета - допомогти вам опанувати нові мови та покращити ваші навички веб-аналітики. Використовуючи англійську, я допоможу студентам покращити їхню граматику, сприймання занять та розмовні навички. Цілого японської, я намагаюся передати не тільки мову, а й культуру вивчення, що дозволяють краще розуміти цю дивовижну країну. Як веб-аналітик, я надаю консультації щодо оптимізації веб-сайтів і допомагаю аналізувати дані для підвищення ефективності онлайн-проектів. Мій підхід - це індивідуальний підхід до кожного учня і клієнта, враховуючи їхні потреби та цілі.

Предмети: Англійська, Японська, Веб-аналітика

Предмети | Про себе | Календар | Відгуки

Умовлені репетитори

Олексій Олексійович Перстов

Я допомагаю студентам вийти від 12 років опанувати мови, покращити їхню граматику та розмовні навички. Мій підхід включає індивідуальні заняття та консультації, орієнтовані на досягнення конкретних цілей учнів і клієнтів.

Японська Англійська Іспанська Веб-аналітика

5 занять 1 відгуків

200 грн/час

Тип заняття: Онлайн | В Учня | У вчителя

Місто: Микопаль/Львівська область

Домашня адреса: Вулиця Велика Морська, 92

Графік свободного часу репетитора

Додати заняту на заняття +

Зображені години, у які є можливість замовити з новими учнями

	Mon 6/3	Tue 6/4	Wed 6/5	Thu 6/6	Fri 6/7	Sat 6/8	Sun 6/9
12am							
1am							
2am							
3am							
4am							
5am							
6am							
7am							
8am							
9am							
10am							
11am							
12pm							
1pm							
2pm							
3pm							
4pm							
5pm							
6pm							
7pm							
8pm							
9pm							
10pm							
11pm							

Додати завдання +

Зм.	Арк.	№ документа	Підпис	Дата	КПІ.ІТ-0221.045440.06.99.КЕ		
Розробив		Морщакін А.Л.			Кресленння вигляду екранних форм	Літера	Маса	Масштаб
Перевірив		Лісовиченко О.І.						
Т. контр.						Аркуш	Аркушів	
Н. контр.		Лищук К.І.			Ведзастосунок для пошуку репетиторів			
Затвердив		Жарікова Е.В.				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-02		

КПІ.ІТ-0221.045440.06.99.КЕ