НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Вебзастосунок для створення та використання цифрових навчальних матеріалів»

Виконав:
студент IV курсу, групи КП-11
Мазний Степан Валерійович _____

Керівник:
доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Тетяна Миколаївна _____

Консультант з нормоконтролю:
доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
доцент кафедри СПіСКС ФПМ, к.т.н. доцент,
Тарасенко-Клятченко Оксана Володимирівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2024 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Мазному Степану Валерійовичу

1. Тема проєкту «Вебзастосунок для створення та використання цифрових навчальних матеріалів», керівник проєкту Заболотня Тетяна Миколаївна, доцент кафедри ПЗКС, к.т.н., доцент, затверджені наказом по університету №1808-С від «29» травня 2025 р.

2. Термін подання студентом проєкту «13» червня 2025 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   – огляд та аналіз наявних рішень;
   – обґрунтування вибору засобів розроблення вебзастосунку;
   – структурно-алгоритмічна організація застосунку;
   – особливості реалізації вебзастосунку.

5. Перелік обов'язкового графічного матеріалу:
   – діаграма взаємозв'язків між сутностями (креслення);
   – блок-схема алгоритму перетворення компонентного дерева у загальний вигляд (креслення);

– діаграма прецедентів розробленого застосунку (плакат);
– граф-модель сценарію «Отримання результатів тесту викладачем» (плакат);
– узагальнена архітектура вебзастосунку (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	14.11.2024	
2.	Розроблення та узгодження технічного завдання	23.11.2024	
3.	Розроблення структури вебзастосунку	18.12.2024	
4.	Підготовка матеріалів першого розділу дипломного проєкту	31.12.2024	
5.	Розроблення дизайну сторінок та графічних елементів	06.02.2025	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2025	
7.	Програмна реалізація вебзастосунку	12.03.2025	
8.	Тестування вебзастосунку	17.03.2025	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2025	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	12.04.2025	
11.	Підготовка графічної частини дипломного проєкту	05.05.2025	
12.	Оформлення документації дипломного проєкту	27.05.2025	

Студент Степан МАЗНИЙ

Керівник проєкту Тетяна ЗАБОЛОТНЯ

АНОТАЦІЯ

Дипломний проєкт присвячено розробці вебзастосунку для створення та використання цифрових навчальних матеріалів у сучасному освітньому середовищі. Основна мета – підвищення якості навчального процесу шляхом спрощення створення, редагування, перегляду та перевірки навчального контенту.

У процесі аналізу предметної галузі було визначено потребу в інструменті, що поєднує зручність для студентів і розширена функціональність для викладачів. Було реалізовано вебзастосунок із клієнтською частиною на Angular, а також з використанням Blazor для роботи з матеріалами в реальному часі. Серверна частина побудована на ASP.NET Core, з використанням баз даних PostgreSQL і MongoDB.

Застосунок дозволяє створювати різні типи навчального контенту, налаштовувати шаблони матеріалів, автоматично перевіряти тести та контролювати доступ. Реалізована система автентифікації на основі JWT та підтримка ролей для розмежування прав користувачів.

Результати тестування підтвердили стабільну роботу системи, надійність логіки перевірки та відповідність функціональним вимогам. Отриманий продукт готовий до використання в освітніх закладах і має потенціал для подальшого вдосконалення та масштабування.

ABSTRACT

The diploma project is dedicated to the development of a web application for creating and using digital learning materials in a modern educational environment. The main goal is to improve the quality of the learning process by simplifying the creation, editing, viewing, and evaluation of educational content.

During the analysis of the subject area, the need for a tool that combines user-friendliness for students with extended functionality for instructors was identified. A web application was implemented with a client-side built on Angular, and Blazor was used to work with content in real time. The server side is developed using ASP.NET Core, with PostgreSQL and MongoDB as the main databases.

The application allows for the creation of various types of learning content, configuration of content templates, automatic test checking, and access control. A JWT-based authentication system and role support were implemented to manage user permissions.

Testing results confirmed the system’s stability, reliability of the evaluation logic, and compliance with functional requirements. The resulting product is ready for use in educational institutions and has strong potential for further improvement and scaling.

ДП.045440-01-90 Вебзастосунок для створення та використання цифрових навчальних матеріалів. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Вебзастосунок для створення	5	
	та використання цифрових		
	навчальних матеріалів.		
	Технічне завдання		
ДП.045440-03-81	Вебзастосунок для створення	67	
	та використання цифрових		
	навчальних матеріалів.		
	Пояснювальна записка		
ДП.045440-04-51	Вебзастосунок для створення	4	
	та використання цифрових		
	навчальних матеріалів.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Вебзастосунок для створення	10	
	та використання цифрових		
	навчальних матеріалів.		
	Керівництво користувача		
ДП.045440-06-99	Вебзастосунок для створення	1	
	та використання цифрових		
	навчальних матеріалів.		
	Діаграма взаємозв’язків між		
	сутностями. ER-діаграма		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Вебзастосунок для створення	1	
	та використання цифрових		
	навчальних матеріалів.		
	Алгоритм перетворення		
	компонентного дерева у		
	загальний вигляд. Блок-схема		
	алгоритму		
ДП.045440-08-98	Вебзастосунок для створення	1	
	та використання цифрових		
	навчальних матеріалів.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ СТВОРЕННЯ ТА ВИКОРИСТАННЯ ЦИФРОВИХ НАВЧАЛЬНИХ МАТЕРІАЛІВ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Степан МАЗНИЙ

2024

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: вебзастосунок для створення та використання цифрових навчальних матеріалів.

Галузь застосування: загальна освіта.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Вебзастосунок для створення та використання цифрових навчальних матеріалів призначений для застосування в освітньому процесі з різних дисциплін. Застосунок дозволяє викладачам створювати та редагувати навчальний контент, налаштовувати тести з автоматичною перевіркою, організовувати доступ до матеріалів, а студентам – зручно переглядати, проходити тести та відстежувати власний прогрес.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Вебзастосунок повинен забезпечувати такі основні функції:

1) можливість створення навчальних матеріалів з текстом, медіа та інтерактивними елементами;
2) редагування матеріалів у візуальному редакторі з автоматичним збереженням;

3) створення тестів із закритими відповідями та автоматичною перевіркою результатів;
4) зручний перегляд контенту студентами через простий інтерфейс;
5) вибір шаблонів сторінок (текст, тест тощо) при створенні матеріалів;
6) реєстрація, авторизація та розподіл ролей користувачів (викладач / студент);
7) доступність системи через веббраузер без потреби встановлення додаткового ПЗ.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   – Діаграма взаємозв’язків між сутностями.
   – Алгоритм перетворення компонентного дерева у загальний вигляд.

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи.....14.11.2024
Розроблення та узгодження технічного завдання.....23.11.2024
Розроблення структури вебзастосунку.....18.12.2024
Розроблення дизайну сторінок та графічних елементів.....06.02.2025
Програмна реалізація вебзастосунку.....12.03.2025

Тестування вебзастосунку.....17.03.2025

Підготовка матеріалів текстової частини проєкту.....12.04.2025

Підготовка матеріалів графічної частини проєкту.....05.05.2025

Оформлення технічної документації проєкту.....27.05.2025

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ СТВОРЕННЯ ТА ВИКОРИСТАННЯ ЦИФРОВИХ НАВЧАЛЬНИХ МАТЕРІАЛІВ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____Степан МАЗНИЙ

2025

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

API (Application Programming Interface) – інтерфейс прикладного програмування, який забезпечує взаємодію між різними програмними компонентами або сервісами. У межах проєкту використовується для обміну даними між frontend-частиною та сервером.

Unit-тестування – процес перевірки окремих функціональних одиниць (модулів, функцій) програмного забезпечення з метою впевненості в їхній коректній роботі незалежно від інших частин системи.

Інтерфейс користувача (UI, User Interface) – візуальна частина системи, через яку користувач взаємодіє з програмою: кнопки, поля введення, повідомлення тощо.

Guid (Globally Unique Identifier) – глобально унікальний ідентифікатор, що часто використовується для позначення записів у базі даних. Дає змогу гарантувати унікальність значень навіть у розподілених системах.

Primary Key (PK) – первинний ключ, унікальний ідентифікатор кожного запису в таблиці бази даних.

Foreign Key (FK) – зовнішній ключ, що встановлює зв’язок між таблицями бази даних. Використовується для підтримки цілісності даних.

DateTime – тип даних, що зберігає значення дати та часу, наприклад, дату створення чи оновлення запису.

JSON (JavaScript Object Notation) – текстовий формат обміну даними, який використовується для зберігання і передачі структурованої інформації між клієнтом і сервером.

ASP.NET Identity / AspNetUsers – вбудована система керування обліковими записами користувачів у фреймворку ASP.NET. Таблиця AspNetUsers містить дані про зареєстрованих користувачів.

Frontend – клієнтська частина вебзастосунку, що відповідає за відображення інтерфейсу користувача й оброблення його взаємодій.

Backend – серверна частина вебзастосунку, що виконує логіку програми, опрацьовує запити, керує базою даних і взаємодіє з API.

CRUD-операції (Create, Read, Update, Delete) – основні дії, які виконуються над даними в системі: створення, читання, оновлення та видалення.

Dropdown (Випадаючий список) – елемент інтерфейсу користувача, що дозволяє вибрати одне значення з попередньо визначеного списку.

Sign in / Log in – процес авторизації користувача у системі за допомогою електронної пошти та пароля.

Sign up / Join Now – процес створення нового облікового запису користувачем у системі.

LMS (Learning Management System) – система управління навчанням. Програмне забезпечення, що використовується для створення, організації та контролю навчального процесу.

JWT (JSON Web Token) – стандарт для передачі інформації між сторонами у вигляді зашифрованого токена. Використовується для автентифікації та авторизації користувачів у вебзастосунках.

Неблокуюча модель вводу-виводу (Non-blocking I/O) – це підхід до оброблення операцій вводу-виводу, за якого виконання програми не зупиняється під час очікування відповіді від джерела даних (наприклад, мережі чи диску).

Dependency Injection (DI) – шаблон проєктування, що передбачає передавання залежностей (об’єктів, від яких залежить клас) ззовні, замість створення їх безпосередньо всередині класу.

Middleware – програмний компонент, який виконує оброблення HTTP-запитів у конвеєрі між запитом клієнта та відповіддю сервера.

Entity Framework (EF) – ORM (Object-Relational Mapper) фреймворк для .NET, що дозволяє працювати з базами даних за допомогою об’єктно-орієнтованого підходу.

SignalR – бібліотека ASP.NET для створення взаємодій в реальному

часі, таких як чати або сповіщення. Вона забезпечує двосторонню комунікацію між сервером і браузером через WebSockets або інші транспортні протоколи.

Identity Server – фреймворк для реалізації служб автентифікації та авторизації, який підтримує протоколи OpenID Connect та OAuth 2.0. Дозволяє централізовано керувати обліковими записами, токенами доступу й авторизаційними політиками в розподілених системах.

SPA (Single Page Application) – тип вебзастосунку, що динамічно перезавантажує лише вміст сторінки без повного перезавантаження всієї HTML-сторінки.

REST API – архітектурний стиль для побудови вебсервісів, де клієнт і сервер взаємодіють через HTTP-запити, а ресурси ідентифікуються за URL.

BSON (Binary JSON) – двійковий формат представлення даних, що є розширенням JSON.

nginx – високопродуктивний вебсервер і зворотній проксі-сервер, що використовується для оброблення HTTP-запитів, балансування навантаження та кешування.

ВСТУП

У сучасному освітньому процесі все більшу роль відіграють цифрові технології, які дають змогу зробити навчання більш доступним, інтерактивним та ефективним. Це є особливо важливим в умовах сьогоднішнього інформаційного суспільства, коли потрібно створювати, поширювати та оновлювати навчальні матеріали відповідно до потреб викладачів та учнів. Традиційні методи, такі як друковані посібники або текстові документи, часто не дозволяють оперативно вносити зміни, не підтримують інтерактивність і мають обмеження у візуалізації складного матеріалу.

Існуючі платформи для розміщення навчального контенту здебільшого орієнтовані на споживання готових матеріалів, а не на їхнє створення. Це створює певні труднощі для викладачів, які хочуть адаптувати або розробляти власні ресурси, використовуючи сучасні цифрові інструменти без необхідності глибоких технічних знань. Крім того, студенти також потребують зручного інтерфейсу для перегляду та взаємодії з такими матеріалами.

Метою даного дипломного проєкту є забезпечити гнучку роботу з цифровими навчальними матеріалами шляхом створення відповідного ПЗ. Розроблений застосунок «StudentPortal» дозволить створювати, редагувати та використовувати цифрові навчальні матеріали у зручному онлайн-середовищі. Щоб досягти поставленої мети, необхідно реалізувати функціональність, яка надасть можливість викладачам створювати навчальний контент, а студентам – легко отримувати доступ до матеріалів.

Вебзастосунок «StudentPortal» сприятиме покращенню навчального процесу завдяки наданню можливості роботи з навчальними матеріалами, гнучкого налаштування способу їхньої подачі, підтримці різних типів контенту (текст, зображення, відео, тести) та інтуїтивному інтерфейсу для обох сторін – викладача та студента. Це дозволить не лише покращити якість

навчання, але й стимулювати розвиток інформаційної обізнаності користувачів.

1. ОГЛЯД ТА АНАЛІЗ НАЯВНИХ РІШЕНЬ

1.1. Аналіз особливостей роботи з цифровими навчальними матеріалами

Цифровізація освіти призвела до суттєвих змін у підходах до створення та використання навчального контенту. Цифрові навчальні матеріали стали важливою складовою сучасного освітнього середовища, забезпечуючи гнучкість у подачі інформації, інтерактивність, а також можливість швидкої адаптації під потреби викладача та студента. Нижче розглянемо детальніше, що собою являють цифрові навчальні матеріали, а також які переваги та недоліки вони мають.

1.1.1. Аналіз особливостей роботи з цифровими навчальними матеріалами

Цифрові навчальні матеріали – це поширювані в цифровому вигляді навчальні матеріали (наприклад, презентація, відео, аудіо, інтерактивні завдання, тест тощо), який містить текст, графічні та мультимедійні елементи і можуть бути інтерактивними [1]. До таких матеріалів належать електронні підручники, інтерактивні презентації, відеолекції, тести, тренажери, симулятори та інші елементи, що використовуються в процесі навчання.

На відміну від традиційних друкованих ресурсів, цифрові матеріали забезпечують вищу гнучкість, можливість інтеграції мультимедійних компонентів та підтримку зворотного зв’язку з учнями.

1.1.2. Переваги використання цифрових ресурсів у навчанні

Використання цифрових технологій у навчальному процесі має низку переваг:

1. Доступність: матеріали можуть бути доступні в режимі онлайн 24/7 з будь-якого пристрою, що підтримує інтернет-з’єднання. Це

значно спрощує навчання в умовах дистанційної або змішаної форми.

2. Мультимедійність: можливість поєднання тексту, зображень, аудіо та відео сприяє кращому засвоєнню інформації різними типами учнів.
3. Інтерактивність: взаємодія з контентом через тести, вправи, інтерактивні елементи дозволяє залучити учня до активного навчального процесу.
4. Актуальність: електронні матеріали можна швидко оновлювати, додаючи нову інформацію, змінюючи структуру або зміст відповідно до навчального плану.
5. Персоналізація: за допомогою цифрових інструментів викладачі можуть адаптувати матеріали під потреби конкретної групи студентів або окремих осіб.

1.1.3. Проблеми, пов'язані з використанням цифрових матеріалів

Попри велику кількість переваг, використання цифрових навчальних матеріалів має і певні виклики:

1. Технічні обмеження: не всі учні або викладачі мають доступ до сучасних пристроїв чи стабільного Інтернету, що створює цифрову нерівність.
2. Проблеми стандартизації: відсутність єдиних форматів або структур для цифрових матеріалів ускладнює їх використання в різних системах або при переході між платформами.
3. Авторські права: створення та поширення електронних матеріалів вимагає дотримання законодавства у сфері інтелектуальної власності.
4. Перевантаження контентом: без чіткої структури та системи навігації цифрові матеріали можуть бути складними для сприйняття та використання.

5. Проблеми зосередженості: у цифровому середовищі студентам важче зберігати концентрацію, адже одночасно з навчальними ресурсами доступні розважальні платформи та соціальні мережі.

1.2. Аналіз наявних програмних рішень

У сучасному цифровому середовищі існує велика кількість програмних продуктів, спрямованих на підтримку освітнього процесу. Ці рішення охоплюють як повноцінні системи управління навчанням (LMS), так і окремі інструменти для створення навчального контенту. Їх активне використання у школах, вишах та позашкільній освіті свідчить про значний попит на цифрові засоби навчання, які дозволяють не лише поширювати інформацію, але й підвищувати її інтерактивність та доступність.

Проте попри різноманіття інструментів, чітко окреслюється низка обмежень у контексті створення індивідуальних цифрових навчальних матеріалів, особливо коли мова йде про зручність використання, адаптацію під конкретні потреби викладача, або ж швидке збирання вмісту за власним структурним і візуальним шаблоном. Існуючі платформи здебільшого орієнтовані на викладання готових курсів, тоді як інструментів-конструкторів, що дозволяли б швидко створити та змінювати унікальні навчальні сторінки, фактично немає.

Цей підрозділ присвячений огляду найбільш поширених цифрових сервісів, які надають можливості для створення, редагування та використання навчального контенту. Після аналізу їх функціональних можливостей та обмежень буде визначено, які саме аспекти не покриваються існуючими рішеннями і що потребує доопрацювання або реалізації в рамках нового вебзастосунку.

1.2.1. Огляд популярних платформ для створення навчального контенту

Серед великої кількості цифрових інструментів, що використовуються

в освітньому процесі, можна виокремити кілька найбільш популярних рішень, які дозволяють викладачам та учням працювати з навчальним контентом в електронному форматі. До них належать такі продукти, як Google Classroom, Moodle, Canva for Education. Розглянемо кожен із них детальніше:

1. Google Classroom – це безкоштовний вебзастосунок, створений компанією Google у 2014 році для підтримки організації навчального процесу. Його основною метою є полегшення взаємодії між викладачами та студентами шляхом централізованого управління завданнями, матеріалами, оцінюванням та зворотним зв’язком. Застосунок тісно інтегрований із сервісами Google Drive, Docs, Sheets, Meet, що забезпечує зручність роботи в єдиному середовищі. Інтерфейс системи мінімалістичний та інтуїтивний, однак можливості налаштування візуального оформлення навчальних матеріалів та структура контенту є досить обмеженими [2].
2. Moodle – це одна з найвідоміших у світі платформ для управління навчанням (LMS), яка розробляється з 2002 року Мартіном Доугіамасом (Martin Dougiamas) та підтримується компанією Moodle Pty Ltd. Платформа є відкритою і безкоштовною, що сприяло її широкому поширенню в університетах і школах по всьому світу. Moodle підтримує створення курсів, додавання тестів, завдань, форумів, відстеження прогресу та багато іншого. Завдяки підтримці модулів і плагінів система є дуже гнучкою, однак водночас потребує налаштування з боку адміністратора, а її інтерфейс може бути складним для нових користувачів [3].
3. Canva for Education – це спеціальна версія популярного графічного редактора Canva, адаптована для освітніх цілей. Основна платформа була запущена у 2013 році, а освітня версія з'явилася близько 2019 року. Розробником є австралійська компанія Canva Pty

Ltd. Сервіс надає користувачам змогу створювати візуально привабливі презентації, інфографіку, постери, буклети тощо, використовуючи зручний інтерфейс із шаблонами та елементами drag-and-drop. Canva особливо корисна у візуалізації навчального контенту, однак не підтримує створення повноцінних курсів із структурою, оцінюванням та інтерактивністю [4].

1.2.2. Порівняння функціональних можливостей

Порівняльний аналіз, наданий в таблиці 1, дозволяє виділити ключові характеристики існуючих рішень.

Таблиця 1

Порівняльний аналіз існуючих рішень

Платформа	Створення контенту	Інтерактивність	Простота використання
Google Classroom	−	Середня	Висока
Moodle	+	Висока	Низька
Canva for Education	+	Середня	Висока

Як видно з таблиці, жодне з рішень не поєднує у собі простоти використання, гнучкості створення навчального контенту та можливості зберігати його у власному форматі в межах одного вебзастосунку.

1.2.3. Обмеження існуючих рішень

Незважаючи на широку функціональність перелічених сервісів, їх використання у ролі універсального конструктора навчальних матеріалів має такі обмеження:

1. Відсутність єдиного простого інтерфейсу для створення власного формату матеріалів – більшість платформ орієнтовані на

використання шаблонів або модулів із фіксованими структурами.

2. Залежність від зовнішніх сервісів – для реалізації окремих функцій потрібно використовувати інші продукти (Google Docs, Forms, PowerPoint тощо).
3. Обмежені можливості персоналізації – не всі сервіси дозволяють гнучко змінювати вигляд або логіку взаємодії з контентом.
4. Технічна складність налаштування – наприклад, для повноцінного використання Moodle потрібні знання адміністрування, що ускладнює його впровадження у малих навчальних закладах.

1.3. Вимоги до застосунку для створення та використання цифрових матеріалів

На основі результатів проведеного аналізу існуючих рішень можна сформулювати ключові вимоги до нового вебзастосунку, який дозволятиме зручно створювати, редагувати та використовувати цифрові навчальні матеріали у власному форматі. Такі вимоги поділяються на функціональні, нефункціональні та архітектурні.

1.3.1. Функціональні вимоги

Функціональні вимоги описують, що саме повинен робити застосунок для виконання своєї основної мети:

1. Створення навчальних матеріалів. Користувач (викладач) повинен мати змогу створювати власні сторінки матеріалів, використовуючи текст, зображення, відео, інтерактивні блоки (наприклад, тести, завдання, коментарі).
2. Редагування та збереження контенту. Необхідна підтримка можливості редагування вже створених матеріалів із збереженням змін. Повинен бути реалізований зручний візуальний редактор.
3. Автоматична перевірка тестів. Система повинна підтримувати створення тестових завдань із закритими відповідями (одна або

кілька правильних), з подальшою автоматичною перевіркою результатів та наданням зворотного зв’язку студенту.

4. Перегляд навчального контенту студентами. Студенти повинні мати доступ до матеріалів через простий і зрозумілий інтерфейс.
5. Шаблони для матеріалів. Користувач повинен мати змогу обирати з кількох шаблонів сторінок (наприклад, тест, просто матеріал) при створенні нового матеріалу, щоб спростити оформлення та структуру контенту.
6. Реєстрація та аутентифікація користувачів. Застосунок має підтримувати реєстрацію, вхід до облікового запису та розподіл ролей (викладач / студент).

1.3.2. Нефункціональні вимоги

Ці вимоги описують якісні характеристики системи:

1. Інтуїтивний інтерфейс. Інтерфейс має бути зрозумілим навіть для користувачів без технічного досвіду. Усі дії повинні виконуватися максимально просто та швидко.
2. Продуктивність і швидкість завантаження. Матеріали повинні завантажуватися швидко, без помітних затримок, навіть при великій кількості контенту.
3. Безпека. Повинна бути реалізована система безпечної аутентифікації, захисту персональних даних, а також резервного копіювання контенту.

1.3.3. Вимоги до адаптивності та масштабованості

Вимоги до системи та інтерфейсу:

1. Адаптивність інтерфейсу. Вебзастосунок має коректно працювати на різних типах пристроїв – комп’ютерах, планшетах, смартфонах. Важливо, щоб користувацький досвід був однаково зручним незалежно від розміру екрана.

2. Масштабованість системи. Архітектура повинна передбачати можливість розширення функціональності у майбутньому: додавання нових типів контенту, підтримку інтеграцій з іншими платформами (наприклад, Google Drive або LMS).
3. Можливість інтеграції з іншими системами. Наявність API або інших засобів обміну даними з зовнішніми сервісами значно підвищить гнучкість застосунку.

1.4. Висновки щодо розділу

У цьому розділі проаналізовано особливості роботи з цифровими навчальними матеріалами, розглянуто найпоширеніші сучасні програмні рішення, які використовуються у сфері електронного навчання, а також сформульовано вимоги до нового вебзастосунку для створення та використання цифрових навчальних матеріалів.

Встановлено, що наявні платформи, хоча й забезпечують базову функціональність для ведення курсів, часто не дозволяють викладачам створювати матеріали у зручному авторському форматі з можливістю гнучкого редагування та інтерактивного наповнення. Значна частина рішень вимагає використання сторонніх сервісів або має складний інтерфейс, що ускладнює їхнє використання у навчальному процесі.

Визначені функціональні та нефункціональні вимоги до розроблюваного вебзастосунку «StudentPortal» дозволяють зробити висновок, що існує потреба у створенні нового інструменту, який би поєднував простоту у використанні, підтримку різноманітного мультимедійного контенту та зручну систему організації й представлення навчальних матеріалів.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ

2.1. Обґрунтування вибору засобів реалізації серверної частини

Серверна частина вебзастосунку відповідає за оброблення бізнес-логіки, взаємодію з базою даних, авторизацію та автентифікацію користувачів, а також за надання клієнтській частині структурованих даних у вигляді API. Надійність, масштабованість та безпека вебсервера мають критичне значення для стабільної роботи застосунку, тому вибір технологій для реалізації серверної частини є ключовим етапом у розробленні будь-якої сучасної вебсистеми. У даному дипломному проєкті для реалізації серверної частини було обрано фреймворк ASP.NET та мову програмування C#.

2.1.1. Основні критерії вибору

При виборі платформи для реалізації серверної логіки було враховано такі основні критерії:

1) продуктивність: здатність системи обробляти велику кількість одночасних запитів з мінімальною затримкою;
2) безпека: підтримка сучасних стандартів захисту даних, включаючи шифрування, автентифікацію та авторизацію;
3) масштабованість: можливість подальшого розширення проєкту без необхідності повного переписування серверної частини;
4) інтеграція з базами даних: сумісність з популярними СКБД, зокрема з реляційними (SQL Server, PostgreSQL);
5) підтримка REST API: зручна реалізація контролерів, маршрутизації та оброблення запитів;
6) наявність документації та підтримки: активна спільнота, офіційна документація та велика кількість навчальних матеріалів;
7) сумісність з хмарними платформами: можливість розгортання в Azure, AWS, Google Cloud тощо.

2.1.2. Огляд можливих технологій

На етапі аналізу було розглянуто кілька сучасних технологій, які широко використовуються для реалізації серверної логіки у вебзастосунках.

1. Node.js (JavaScript/TypeScript) – популярна платформа, орієнтована на оброблення великої кількості одночасних з'єднань. Має неблокуючу модель вводу-виводу та багату екосистему пакетів. Часто використовується в проєктах із високими вимогами до швидкості відповіді [5].
2. Django (Python) – фреймворк з великою кількістю вбудованих інструментів для побудови CRUD-інтерфейсів, автентифікації та адміністрування. Підходить для швидкого розроблення, але поступається в продуктивності компільованим мовам [6].
3. Spring Boot (Java) – потужний фреймворк із широкими можливостями конфігурації, добре підходить для масштабних корпоративних рішень. Проте вимагає більше ресурсів та має вищий поріг входу [7].
4. ASP.NET Core (C#) – кросплатформенна, високопродуктивна технологія від Microsoft, яка поєднує гнучкість у конфігурації, зручність розроблення REST API, високу продуктивність та глибоку інтеграцію з іншими інструментами екосистеми .NET. Підтримує сучасні принципи розроблення – Dependency Injection, Middleware, Entity Framework та інші [8].

2.1.3. Результати проведеного аналізу

Після аналізу вищезазначених рішень було прийнято рішення на користь використання ASP.NET Core. Цей вибір обумовлений низкою важливих факторів:

1. ASP.NET Core забезпечує високу продуктивність завдяки компіляції коду у нативний машинний код та оптимізації на рівні CLR.

2. Екосистема .NET є кросплатформенною, що дозволяє запускати застосунок на Windows, Linux або в контейнерах Docker.
3. Вбудована підтримка автентифікації та авторизації, а також можливість легкої інтеграції з JWT, OAuth2, Identity Server тощо.
4. Застосування архітектури MVC/Web API дозволяє створювати структурований та підтримуваний код.
5. Широка підтримка хмарного середовища Azure відкриває перспективи для подальшого масштабування та розгортання.

Таким чином, ASP.NET Core з C# повністю відповідає поставленим вимогам до серверної частини вебзастосунку «StudentPortal» та забезпечує надійну основу для реалізації необхідної функціональності.

2.2. Обґрунтування вибору засобів реалізації клієнтської частини

Клієнтська частина є незамінним елементом будь-якого вебзастосунка, оскільки саме вона забезпечує взаємодію користувача з усіма функціональними можливостями системи. У випадку вебзастосунку «StudentPortal», клієнтський інтерфейс має бути не лише зручним і інтуїтивно зрозумілим, але й технічно гнучким, щоб ефективно реалізовувати різноманітні підходи до роботи з цифровими навчальними матеріалами.

Особливість цього проєкту полягає в тому, що вебінтерфейс повинен одночасно виконувати кілька різнопланових завдань: презентувати інформацію користувачеві у вигляді публічних сторінок, дозволяти зручно переглядати створені матеріали та забезпечити максимально інтерактивне та швидке середовище для їх редагування. Через це було вирішено реалізувати клієнтську частину з використанням комбінації технологій: Angular, Blazor Server і Blazor WebAssembly. Кожна з них виконує свою чітко визначену функцію, покращуючи окремі частини користувацького досвіду.

2.2.1. Основні критерії вибору

При виборі інструментів реалізації клієнтської частини було визначено низку ключових критеріїв, які мають вирішальний вплив на якість застосунку:

1) продуктивність: клієнтська частина має швидко реагувати на дії користувача та не створювати значного навантаження на сервер;
2) інтерактивність: особливо важлива під час редагування навчальних матеріалів, де потрібна динамічна робота з контентом, зміни стилів, структури, вкладеності тощо;
3) можливість розділення обов'язків між модулями: для кращої гнучкості коду і спрощення розроблення;
4) сумісність із серверною частиною на ASP.NET: для полегшення інтеграції логіки та повторного використання моделей і структур;
5) масштабованість: технологія повинна дозволяти в подальшому легко розширювати функціональність інтерфейсу без суттєвої переробки;
6) уніфікація технологій: бажано мінімізувати кількість мов програмування і фреймворків для зменшення складності проєкту;
7) гнучкість дизайну: важливо забезпечити адаптивність під мобільні пристрої, планшети та ПК;
8) можливість автономної роботи: у випадку з редагуванням матеріалів бажано, щоб зміни не блокувалися мережею, а частково виконувалися на клієнті.

2.2.2. Огляд можливих технологій

Для реалізації інтерфейсу розглядалися кілька сучасних технологій:

1. React – популярна бібліотека для створення UI-компонентів, з багатою екосистемою, підтримкою реактивного оновлення інтерфейсу та великою спільнотою. Вона має хорошу продуктивність, однак не є повноцінним фреймворком, тому для

багатьох завдань потребує додаткових рішень (наприклад, маршрутизації, керування станом) [9].

2. Vue.js – легкий фреймворк з простим API та потужною системою компонентів. Він добре підходить для невеликих проєктів, проте для комплексних корпоративних систем поступається в структурованості Angular або Blazor [10].
3. Angular – повноцінний фреймворк від Google, що підтримує двосторонню прив'язку даних, інжекцію залежностей, модульність, систему шаблонів, маршрутизацію та формальну структуру коду. Це дозволяє створити масштабовану, організовану архітектуру, що легко підтримується. Angular – один з найбільш придатних варіантів для клієнтсько-складного застосунку з великою кількістю компонентів [11].
4. Blazor Server – підхід до побудови вебінтерфейсу, де вся логіка виконується на сервері, а браузер лише відображає результат за допомогою SignalR-з'єднання. Перевагою є можливість писати інтерфейс на тій самій мові (C#), що й сервер, та мінімізувати код на JavaScript. Також забезпечується централізоване управління станом [12].
5. Blazor WebAssembly – альтернатива Blazor Server, яка дозволяє завантажити C#-код у браузер та виконувати логіку безпосередньо на клієнті. Це значно зменшує навантаження на сервер і дозволяє реалізувати офлайн-підтримку або більш гнучке реагування на події користувача [12].

2.2.3. Результати проведеного аналізу

Розглянувши переваги та обмеження кожної з технологій, було обрано комбінований підхід, який дозволяє максимально ефективно реалізувати різні частини інтерфейсу:

1. Angular використано для створення основного каркаса

вебзастосунку. Зокрема, цей фреймворк реалізує головну сторінку, маршрутизацію, форми реєстрації та авторизації, інтерфейс керування профілем користувача та сторінки налаштувань. Його архітектура дозволяє створити масштабовану SPA з чітко розділеними модулями та хорошою інтеграцією з REST API.

2. Blazor Server застосовується для реалізації перегляду навчальних матеріалів. Це дає можливість максимально повторно використовувати код на C# між сервером і клієнтом, уникнути зайвих API-викликів і забезпечити повний контроль над відображенням матеріалів. Крім того, даний підхід дозволяє обробляти стан клієнтської частини у реальному часі, що критично для інтерактивного контенту.
3. Blazor WebAssembly використовується у модулі редагування цифрових навчальних матеріалів. Завдяки WebAssembly, редактор працює без затримок і оновлень від сервера, що забезпечує миттєву реакцію на дії користувача – зміни тексту, структури, макету або мультимедійного вмісту. Ця частина застосунку найбільш ресурсомістка і вимагає складної логіки на клієнті, тому саме Blazor Wasm став оптимальним рішенням.

Такий багаторівневий підхід дав змогу поєднати переваги JavaScript-фреймворку (Angular) для загального інтерфейсу з можливостями C#-орієнтованих технологій (Blazor) для роботи з матеріалами. Це дозволяє оптимізувати час розроблення, повторно використовувати моделі та структури, а також забезпечити стабільну взаємодію клієнта й сервера без необхідності дублювання логіки.

2.3. Обґрунтування вибору систем керування базами даних (СКБД)

У сучасних вебзастосунках зберігання, оброблення та ефективне управління даними є одним із ключових компонентів архітектури. Залежно

від типу інформації, що зберігається, можуть застосовуватись різні моделі даних – реляційна, документна, графова тощо. У дипломному проєкті «StudentPortal», який поєднує традиційні елементи користувацьких облікових записів, ролей, структурованих налаштувань та, одночасно, складних динамічних навчальних матеріалів, було прийнято рішення використовувати дві СКБД: реляційну PostgreSQL як основну та документно-орієнтовану MongoDB для зберігання контенту цифрових матеріалів.

Таке поєднання дозволяє максимально ефективно працювати як з традиційними структурованими даними, так і з динамічним, змінним за структурою вмістом навчальних ресурсів, зокрема текстів, блоків, вкладених елементів, зображень, списків та іншого навчального контенту.

2.3.1. Основні критерії вибору

Вибір систем керування базами даних здійснювався на основі таких критеріїв:

1) надійність і стабільність роботи;
2) широка підтримка та документація;
3) масштабованість для майбутнього розширення функціональності;
4) продуктивність при виконанні складних запитів;
5) гнучкість при зберіганні даних різної структури;
6) можливість ефективної взаємодії з технологіями ASP.NET та C#;
7) підтримка транзакцій та цілісності даних;
8) можливість резервного копіювання та відновлення даних.

2.3.2. Обґрунтування вибору PostgreSQL

PostgreSQL – об'єктно-реляційна система керування базами даних з відкритим вихідним кодом, яка підтримує стандарт SQL та розширення з великою кількістю можливостей [13]. У межах проєкту вона використовується як основна СКБД для зберігання даних користувачів

(акаунти, ролі, сесії, дані користувачів);

Переваги використання PostgreSQL у даному контексті:

1) повна сумісність з Entity Framework Core у середовищі .NET;
2) висока продуктивність при великій кількості одночасних з'єднань;
3) безплатний, активно підтримується спільнотою та постійно оновлюється;
4) зручне масштабування при потребі розширення даних.

Завдяки PostgreSQL забезпечується цілісність критично важливих даних, строгий контроль доступу, транзакційність та ефективна аналітика через складні SQL-запити.

2.3.3. Обґрунтування вибору MongoDB

Окрім реляційної СКБД, для реалізації повної функціональності вебзастосунку «StudentPortal» було обрано також MongoDB – сучасну документно-орієнтовану систему керування базами даних [14]. Її використання зумовлене архітектурною необхідністю зберігати дані, структура яких не є фіксованою та передбачає вкладеність, змінність і деревоподібну організацію.

Навчальні матеріали, які створюються та редагуються в середовищі застосунку, мають складну внутрішню будову, що може включати численні рівні вкладеності, різнорідні елементи, а також гнучкі зв'язки між ними. У таких випадках традиційна реляційна модель ускладнює реалізацію структури даних, потребуючи значної нормалізації та великої кількості пов'язаних таблиць. На противагу цьому, документна модель MongoDB дозволяє безпосередньо зберігати складні об'єкти у вигляді дерев з довільною глибиною та структурою.

Саме ця властивість MongoDB – можливість ефективного зберігання деревоподібних об'єктів у форматі BSON – робить її влучним вибором для зберігання вмісту цифрових матеріалів у рамках вебзастосунку. Вона дозволяє уникнути надлишкової складності на рівні схеми, прискорює

доступ до вкладених елементів і полегшує реалізацію функціональнсоті редагування та збереження контенту без необхідності постійного узгодження з жорсткою структурою таблиць.

MongoDB також добре інтегрується з технологіями .NET, має офіційну бібліотеку для C# та підтримує розширену індексацію, що в майбутньому дозволить реалізувати гнучкий пошук і фільтрацію навчальних об’єктів.

Таким чином, обрання MongoDB як додаткової СКБД забезпечує ефективну реалізацію гнучкої та масштабованої моделі зберігання, необхідної для роботи з цифровими навчальними матеріалами, які мають ієрархічну структуру та динамічний вміст.

2.3.4. Результати проведеного аналізу

Комбіноване використання PostgreSQL і MongoDB дозволяє досягти високої продуктивності, масштабованості та зручності у роботі як зі структурованими, так і з динамічними даними. PostgreSQL гарантує надійність для критичних бізнес-даних, тоді як MongoDB дає гнучкість у побудові та редагуванні цифрових навчальних матеріалів, які постійно змінюються в процесі розроблення та використання.

Таке розділення відповідальності між двома системами зберігання забезпечує оптимальну архітектуру з точки зору продуктивності, підтримки та майбутнього розширення функціональності застосунку.

2.4. Висновки до розділу

У цьому розділі було здійснено аналіз та обґрунтування вибору технологій, які використовуються для реалізації вебзастосунку «StudentPortal». Розгляд проводився окремо для серверної частини, клієнтської частини та систем керування базами даних, що дозволило врахувати особливості кожного компонента системи.

Для розроблення серверної частини обрано платформу ASP.NET з мовою програмування C#, що забезпечує високу продуктивність, підтримку

сучасних стандартів веброзроблення та зручну інтеграцію з іншими технологіями .NET-екосистеми.

Клієнтську частину побудовано з використанням Angular, Blazor Server та Blazor WebAssembly. Такий розподіл дає змогу ефективно організувати інтерфейс користувача, розмежовуючи загальну функціональність (реєстрація, налаштування, навігація) та спеціалізовану роботу з навчальними матеріалами, яка вимагає як інтерактивності, так і високої продуктивності.

Для управління даними застосовано комбінацію двох СКБД: PostgreSQL як основної реляційної бази для зберігання структурованої інформації (користувачі, ролі, доступи), та MongoDB як документно-орієнтованої системи для роботи з деревоподібними структурами навчальних матеріалів.

Загалом, вибір інструментів розроблення базується на критеріях надійності, масштабованості, продуктивності та відповідності функціональним вимогам до системи. Такий підхід дозволяє створити ефективний, стабільний і гнучкий вебзастосунок для створення та використання цифрових навчальних матеріалів.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ ВЕБЗАСТОСУНКУ

3.1. Формулювання вимог до програмного забезпечення

Розроблення програмного забезпечення є складним інженерним процесом, який передбачає чітке визначення вимог до системи ще на початкових етапах життєвого циклу проєкту. Саме від якості та повноти сформульованих вимог залежить успішність подальших етапів проєктування, реалізації, тестування та впровадження. Наявність узгодженого переліку вимог дає змогу забезпечити відповідність між очікуваннями кінцевих користувачів, технічними можливостями системи та її фактичною функціональністю.

У рамках дипломного проєкту розробляється вебзастосунок StudentPortal, призначений для підтримки процесів створення, редагування, перегляду та організації цифрових навчальних матеріалів.

У даному підпункті здійснюється формалізація вимог до програмного забезпечення із поділом їх на функціональні та нефункціональні. Функціональні вимоги (табл. 3.1) описують основні можливості системи. Нефункціональні вимоги (табл. 3.2) охоплюють технічні та експлуатаційні характеристики застосунку: продуктивність, надійність, масштабованість, безпеку та сумісність із сучасними веббраузерами.

Таблиця 3.1

Реєстр функціональних вимог до програмного продукту

Код вимоги	Зміст вимоги
F-1	З головної сторінки користувач повинен мати змогу здійснити авторизацію в особистому кабінеті.
F-2	Після успішної авторизації користувач має бути автоматично перенаправлений на сторінку зі своїми навчальними матеріалами.

Продовження табл. 3.1

F-3	На сторінці реєстрації користувач повинен мати можливість швидко перейти до форми входу, і навпаки.
F-4	У викладача має бути можливість здійснювати пошук студентів за іменем або електронною поштою.
F-5	Викладач повинен мати доступ до деталізованих відповідей кожного студента шляхом натискання кнопки «Details».
F-6	На сторінці з результатами тесту має бути передбачено кнопку «Grade» для ручного оцінювання відповідей викладачем.
F-7	Форма реєстрації повинна здійснювати валідацію формату email та перевірку складності пароля перед надсиланням даних.
F-8	У процесі створення навчального матеріалу користувач повинен бачити інтерактивний попередній перегляд результату.
F-9	Користувач повинен мати змогу копіювати окремі компоненти матеріалів та вставляти їх повторно.
F-10	Під час створення нового матеріалу користувач має мати доступ до вибору з-поміж заздалегідь підготовлених шаблонів.
F-11	Застосунок повинний зберігати поточний прогрес проходження тесту навіть у разі оновлення або випадкового перезавантаження сторінки.
F-12	Результати проходження тесту повинні зберігатися в базі даних після завершення тестування.
F-13	Система повинна забезпечувати логування подій, пов’язаних зі створенням і редагуванням матеріалів.
F-14	Усі користувачі повинні мати змогу змінювати особисті дані (ім’я, email, пароль) у профілі.

Продовження табл. 3.1

F-15	Система повинна надавати можливість переглядати матеріали учням

Таблиця 3.2

Реєстр нефункціональних вимог до програмного продукту

Код вимоги	Зміст вимоги
NF-1	Інтерфейс користувача має відповідати принципам доступності, зокрема стандартам контрастності та читабельності для людей з порушенням зору.
NF-2	Навчальні матеріали повинні зберігатися у вигляді деревоподібних об’єктів із підтримкою версійності кожного компонента.
NF-3	Усі паролі користувачів повинні зберігатися у базі даних у вигляді криптографічного хешу з використанням сучасних алгоритмів шифрування.
NF-4	Усі основні функціональні модулі системи мають проходити модульне тестування на кожному етапі розроблення.
NF-5	Система має забезпечувати високу відмовостійкість та зберігати працездатність при високому навантаженні.
NF-6	Час відгуку серверної частини на типові запити не повинен перевищувати 500 мс у більшості випадків.
NF-7	Програмне забезпечення повинно коректно працювати в актуальних версіях популярних веббраузерів (Chrome, Firefox, Edge).
NF-8	Вебзастосунок повинен бути масштабованим і підтримувати розгортання у хмарному середовищі (наприклад, через Docker).

3.2. Сценарії використання застосунку

Для ілюстрації ключових взаємодій користувачів із системою

доцільно використати діаграму прецедентів (Use Case Diagram). Вона відображає основні ролі – студента та викладача, їхню взаємодію з основними функціями застосунку: реєстрація, вхід у систему, створення та редагування навчальних матеріалів, проходження тестів, оцінювання результатів тощо.

Окрім цього, з метою деталізації логіки виконання окремих операцій, подано діаграму послідовностей (Sequence Diagram), яка демонструє динаміку подій під час одного з ключових процесів – отримання результатів тесту викладачем. Це дозволяє краще зрозуміти внутрішню структуру взаємодії компонентів системи та забезпечити її цілісність і надійність.

Наведемо діаграми на рис. 3.1 і рис. 3.2 відповідно.

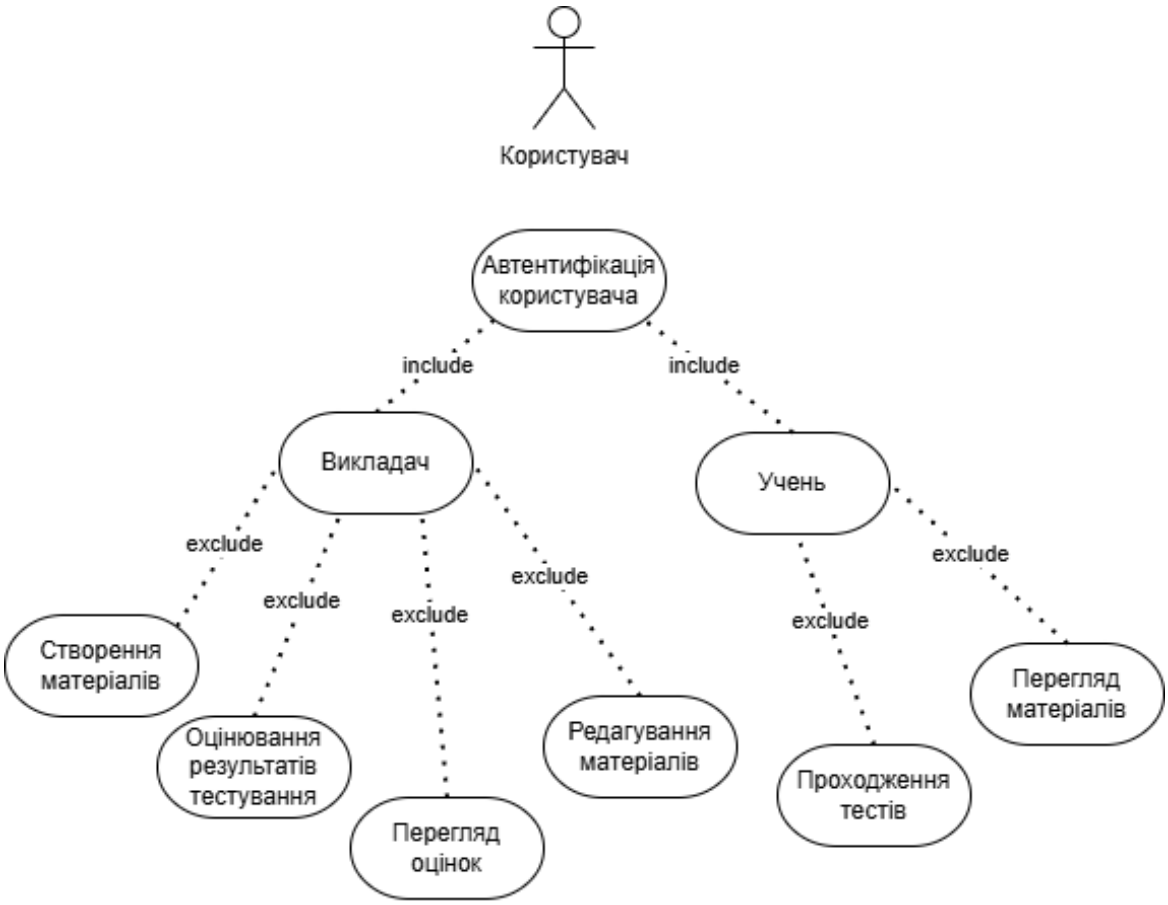


Рис. 3.1. Діаграма прецедентів розробленого застосунку

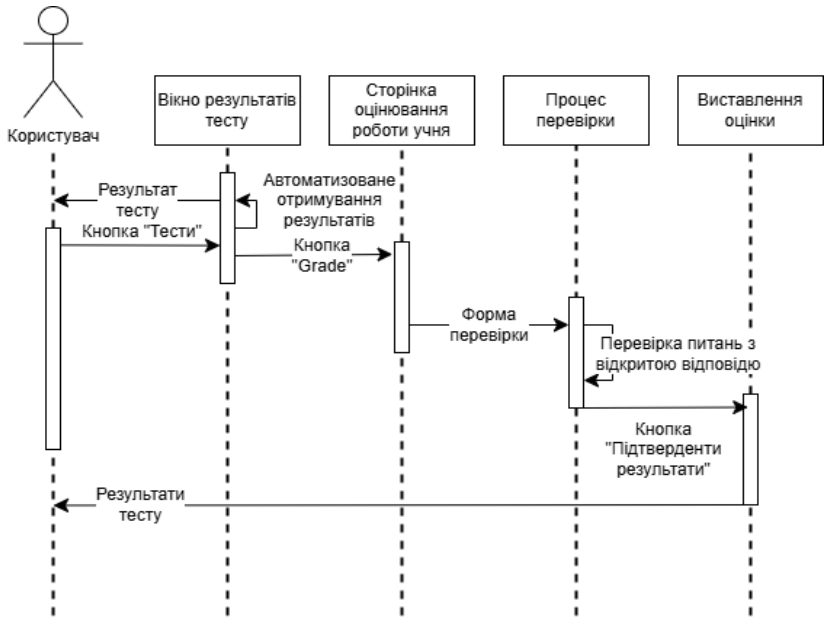


Рис. 3.2. Граф-модель сценарію «Отримання результатів тесту викладачем»

Таблиця 3.3

Таблиця сценаріїв

№	Назва та код сценарію	Опис сценарію
1	Код сценарію: А-1 Назва: Створення нового цифрового навчального матеріалу Актори: Викладач, frontend-розробник, backend-розробник Пріоритет: Високий Частота використання: Під час кожного створення нового навчального матеріалу	Викладач створює новий цифровий матеріал, який надалі може бути доступним для студентів або інших користувачів. Попередні умови: Користувач авторизований у системі та має роль викладача. Подальші умови: Матеріал збережено на сервері та доступний для перегляду. Нормальна послідовність дій: 1) Користувач переходить на вкладку “Мої матеріали” 2) Обирає опцію створення нового матеріалу 3) Вибирає відповідний шаблон 4) Заповнює вміст у редакторі 5) Зберігає матеріал

Продовження табл. 3.3

		6) Поширює матеріал за допомогою згенерованого посилання Винятки: Технічні помилки під час збереження Бізнес-правила: Інтерфейс має бути інтуїтивно зрозумілим, надавати функції автозбереження, відміни дій, копіювання та повторного використання шаблонів
2	Код сценарію: А-2 Назва: Перевірка результатів тестування студентів Актори: Студент, викладач, frontend-розробник Пріоритет: Високий Частота використання: Після кожного завершеного тесту	Викладач здійснює перевірку результатів тесту, включаючи ручне оцінювання, якщо тест містить відкриті питання. Попередні умови: Авторизація викладача і студента у системі. Студент пройшов тест. Подальші умови: Отримано остаточну оцінку студента Нормальна послідовність дій: 1) Студент проходить тест 2) Викладач відкриває результати 3) Система автоматично оцінює частину тесту 4) Якщо є відкриті запитання, викладач натискає "Оцінити" 5) Вручну оцінює відповіді 6) Отримує загальну оцінку Винятки: Тест не завершено повністю, неможливість оцінити Бізнес-правила: Відкриті запитання перевіряються виключно вручну
3	Код сценарію: А-3 Назва: Авторизація користувача	Користувач вводить логін і пароль, щоб отримати доступ до системи. Попередні умови: Відсутність

Продовження табл. 3.3

	Актори: Користувач, backend-розробник, frontend-розробник Пріоритет: Високий Частота використання: Регулярно	активної сесії Подальші умови: Отримано авторизований доступ до системи Нормальна послідовність дій: 1) Користувач відкриває сторінку входу 2) Вводить логін і пароль 3) Підтверджує вхід 4) Отримує доступ до функцій системи Альтернативна послідовність дій: 1) Введення неправильних облікових даних 2) повідомлення про помилку Винятки: Збій з'єднання з сервером або базою даних Бізнес-правила: Всі дані передаються через захищене з’єднання (HTTPS), автентифікація реалізована через JWT
4	Код сценарію: А-4 Назва: Тестування роботи програмного забезпечення Актори: Тестувальник, frontend-розробник, backend-розробник Пріоритет: Середній Частота використання: Після кожної ітерації розроблення	Виявлення та усунення дефектів у роботі системи через тестування Попередні умови: Завершення одного з етапів розроблення Подальші умови: Система працює коректно Нормальна послідовність дій: 1) Запуск юніт-тестів 2) Проведення ручного тестування інтерфейсу 3) Виявлення проблем 4) Надсилання звітів розробникам

Продовження табл. 3.3

		5) Виправлення та повторне тестування Винятки: Невідповідність між оточенням тестування і продакшн Бізнес-правила: Необхідність відповідності очікуваній поведінці

3.3. Опис архітектури програмного забезпечення

Архітектура програмного забезпечення у рамках даного проєкту побудована з використанням сучасних підходів, орієнтованих на модульність, масштабованість та гнучкість. Основною концепцією, яка лежить в основі архітектури, є розділення клієнтської та серверної частин з чітким визначенням зон відповідальності, що дозволяє ефективно підтримувати та розширювати систему.

Система реалізована за принципом клієнт-серверної взаємодії, де клієнтська частина (frontend) функціонує як SPA (Single Page Application) та взаємодіє із серверною частиною (backend) через REST API. Такий підхід забезпечує мінімальне навантаження на мережу, високу швидкість відгуку інтерфейсу та гнучкість в оновленні клієнтської частини незалежно від серверної.

Клієнтська частина побудована з використанням Angular або Blazor (залежно від модуля), обидва дозволяють реалізовувати компонентно-орієнтований підхід до побудови інтерфейсу. Компоненти інкапсулюють логіку та візуальне представлення, що сприяє їх повторному використанню та зменшенню складності при масштабуванні інтерфейсу.

Серверна частина побудована з урахуванням модульної архітектури, де кожен модуль відповідає за окрему функціональність – наприклад, автентифікацію, управління користувачами, роботу з навчальними матеріалами тощо. Такий підхід дозволяє реалізувати чисту архітектуру,

забезпечивши мінімальну залежність між модулями та легкість у тестуванні й розширенні.

Комунікація між модулями здійснюється за допомогою впровадження залежностей (Dependency Injection), що дозволяє легко замінювати реалізації сервісів без змін у їх використанні. Крім того, в обробці запитів бере участь Middleware-рівень, який відповідає за обробку крос-функціональних задач, таких як автентифікація, логування, обробка помилок тощо.

Важливим елементом архітектури є використання reverse proxy-сервера, який виконує роль проміжного шару між клієнтами та серверними застосунками. Reverse proxy забезпечує маршрутизацію запитів до відповідних сервісів, балансування навантаження, кешування статичних ресурсів та захист від деяких типів атак. Такий підхід також дозволяє легко розгортати кілька сервісів одночасно та централізовано управляти точкою входу до системи.

Для роботи з базою даних використовується ORM-технологія Entity Framework, що дозволяє працювати з даними у вигляді об’єктів, спрощуючи доступ до даних, реалізацію запитів та забезпечуючи типобезпечність.

У рамках архітектури також було застосовано контейнеризацію за допомогою технології Docker, що дозволило ізолювати окремі компоненти системи (клієнтську частину, серверну логіку, базу даних) у самостійні контейнери. Це значно спрощує процес розгортання, забезпечує однакове середовище для розробки та продакшену, а також полегшує масштабування та оновлення окремих частин системи. Завдяки використанню Docker Compose була реалізована можливість запуску всієї інфраструктури за допомогою одного конфігураційного файлу, що спрощує як локальну розробку, так і розгортання на сервері.

Завдяки застосованим архітектурним підходам, програмне забезпечення легко масштабувати, тестувати та підтримувати, а також адаптувати під майбутні вимоги без суттєвих змін у структурі системи.

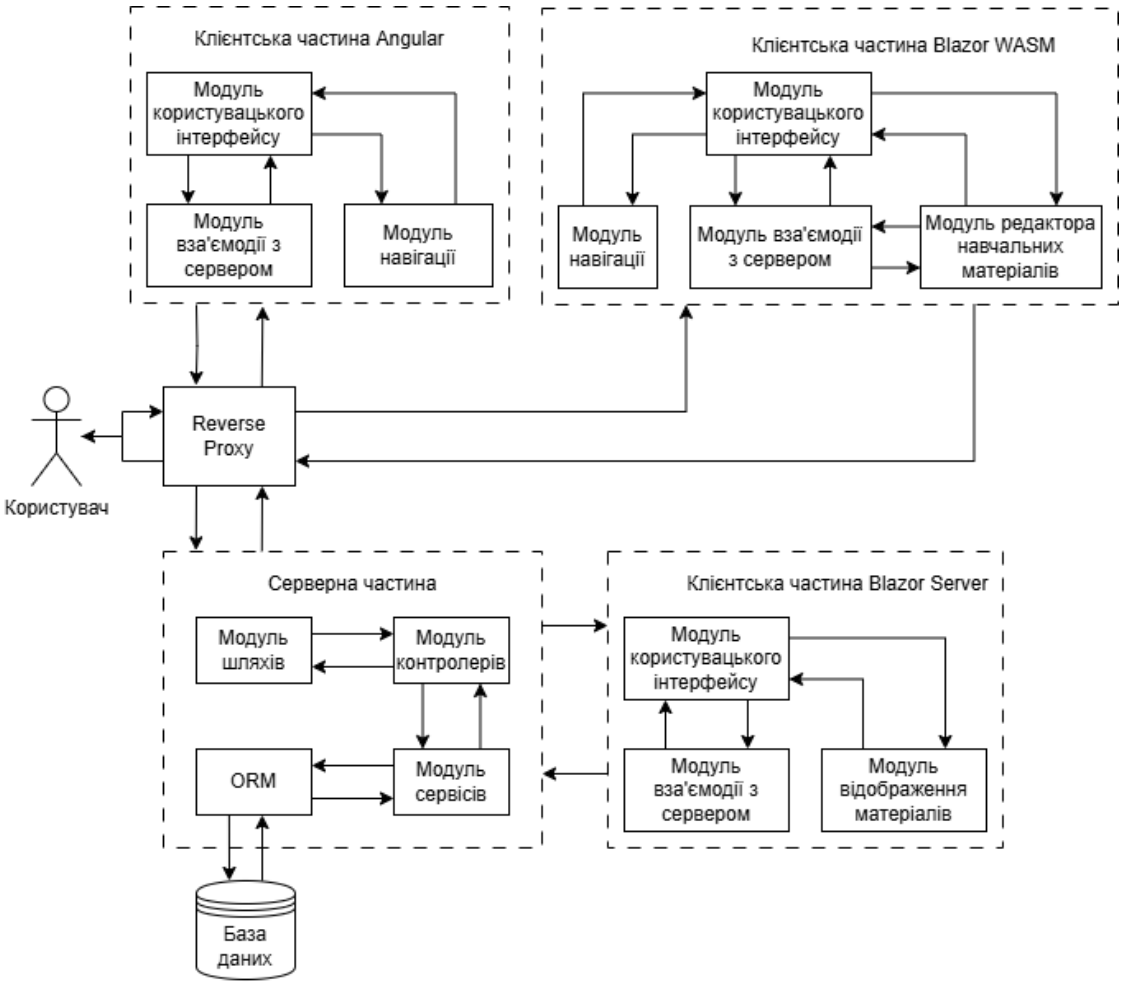


Рис. 3.3. Узагальнена архітектура вебзастосунку

3.4. Опис моделі даних застосунку

У цьому підпункті наведено загальний опис структури баз даних, представлено зв’язки та діаграму сутностей (рис 3.4), а також подано коротку характеристику ключових сутностей, які використовуються в системі для забезпечення основної функціональності.

Опишемо зв’язки, що були створені в процесі створення вебзастосунку:

AspNetUsers – AspNetUserRoles

Тип зв’язку: один-до-багатьох. Пояснення: Один користувач може мати декілька ролей, які пов’язані з ним через таблицю AspNetUserRoles.

AspNetRoles – AspNetUserRoles

Тип зв’язку: один-до-багатьох. Пояснення: Одна роль може бути

призначена багатьом користувачам.

AspNetUsers – AspNetUserClaims

Тип зв’язку: один-до-багатьох. Пояснення: Один користувач може мати декілька клеймів (атрибутів безпеки).

AspNetUsers – AspNetUserLogins

Тип зв’язку: один-до-багатьох. Пояснення: Один користувач може здійснювати вхід через декілька зовнішніх провайдерів.

AspNetUsers – AspNetUserTokens

Тип зв’язку: один-до-багатьох. Пояснення: Один користувач може мати декілька токенів, що використовуються для автентифікації або збереження сеансів.

AspNetUsers – Notification

Тип зв’язку: один-до-багатьох. Пояснення: Один користувач може отримати багато повідомлень, кожне з яких має посилання на ідентифікатор користувача.

AspNetUsers – Page (MongoDB)

Тип зв’язку: один-до-багатьох. Пояснення: Один користувач може бути власником багатьох цифрових навчальних матеріалів, що зберігаються у колекції Page. Зв’язок встановлюється через поле OwnerId.

AspNetUsers – QuizData

Тип зв’язку: один-до-багатьох. Пояснення: Один користувач може проходити декілька тестів, кожен з яких зберігається як окремий запис у таблиці QuizData.

QuizData – QuizResults

Тип зв’язку: один-до-багатьох. Пояснення: Для кожного питання створюється окремий запис у таблиці QuizResults, що зберігає деталі питання, відповідь, тип питання, максимально можливу оцінку та фактичний бал користувача.

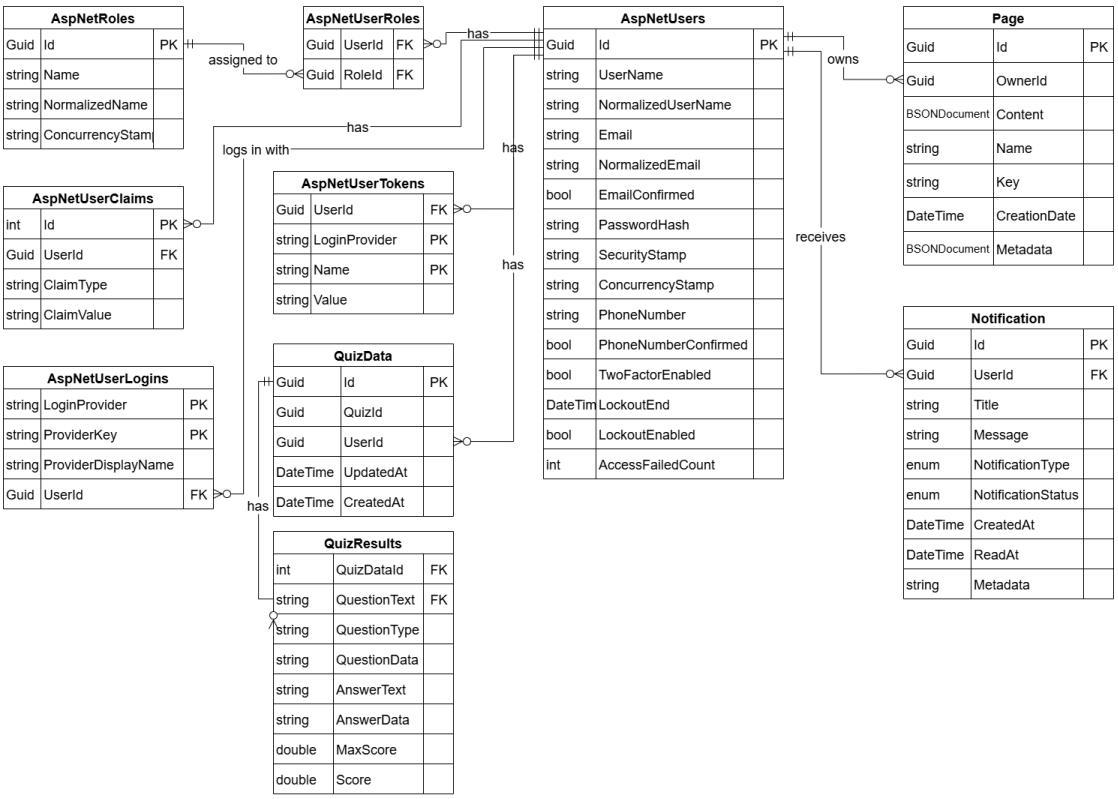


Рис. 3.4. Діаграма взаємозв’язків між сутностями

Таблиця 3.4

Таблиця AspNetUsers

Поле	Тип	Опис
Id	Guid	Унікальний ідентифікатор користувача
UserName	String	Ім’я користувача
NormalizedUserName	String	Уніфіковане ім’я користувача
Email	String	Електронна пошта
NormalizedEmail	string	Уніфікована електронна пошта
EmailConfirmed	bool	Статус підтвердження пошти
PasswordHash	string	Хеш пароля

Продовження табл. 3.4

PhoneNumber	string	Номер телефону
PhoneNumberConfirmed	bool	Статус підтвердження телефону
TwoFactorEnabled	bool	Двофакторна автентифікація
LockoutEnd	DateTime	Дата завершення блокування
LockoutEnabled	bool	Чи дозволено блокування
AccessFailedCount	int	Кількість невдалих входів

Таблиця 3.5

Таблиця AspNetRoles

Поле	Тип	Опис
Id	Guid	Унікальний ідентифікатор ролі
Name	string	Назва ролі
NormalizedName	string	Уніфікована назва ролі
ConcurrencyStamp	string	Ідентифікатор версії для паралельності

Таблиця 3.6

Таблиця AspNetUserRoles

Поле	Тип	Опис
UserId	Guid	Посилання на користувача
RoleId	Guid	Посилання на роль

Таблиця 3.7

Таблиця AspNetUserLogins

Поле	Тип	Опис
LoginProvider	string	Провайдер автентифікації
ProviderKey	string	Ключ від провайдера
ProviderDisplayName	string	Назва провайдера
UserId	Guid	Посилання на користувача

Таблиця 3.8

Таблиця AspNetUserTokens

Поле	Тип	Опис
UserId	Guid	Посилання на користувача
LoginProvider	string	Назва провайдера
Name	string	Назва токена
Value	string	Значення токена

Таблиця 3.9

Таблиця Notification

Поле	Тип	Опис
Id	Guid	Унікальний ідентифікатор повідомлення
UserId	Guid	Користувач, якому належить повідомлення
Title	string	Заголовок повідомлення

Продовження табл. 3.10

Message	string	Основний текст
NotificationType	int	Тип повідомлення
NotificationStatus	int	Статус
CreatedAt	DateTime	Дата створення
ReadAt	DateTime	Дата прочитання
Metadata	string (JSON)	Додаткові дані у форматі JSON

Таблиця 3.11

Таблиця Page (MongoDB)

Поле	Тип	Опис
Id	Guid	Унікальний ідентифікатор сторінки
OwnerId	Guid	Ідентифікатор користувача, власника
Content	BSONDocument	Документ зі структурою матеріалу
Name	string	Назва сторінки
Key	string	Унікальний ключ доступу або ідентифікації
CreationDate	DateTime	Дата створення
Metadata	BSONDocument	Додаткова інформація про сторінку

Таблиця 3.12

Таблиця QuizData

Поле	Тип	Опис
Id	Guid	Унікальний ідентифікатор запису

Продовження табл. 3.12

QuizId	Guid	Ідентифікатор тесту
UserId	Guid	Ідентифікатор користувача, власника
UpdatedAt	DateTime	Дата та час останнього оновлення запису.
CreatedAt	DateTime	Дата та час створення запису.

Таблиця 3.13

Таблиця QuizResults

Поле	Тип	Опис
QuizDataId	int	Зовнішній ключ. Посилання на запис у таблиці QuizData.
QuestionText	string	Текст питання. Використовується також як частина ключа.
QuestionType	string	Тип питання (наприклад, вибір однієї відповіді, введення тексту тощо).
QuestionData	string	Додаткові дані або параметри питання у форматі JSON або тексту.
AnswerText	string	Відповідь користувача у вигляді тексту.
AnswerData	string	Додаткові дані відповіді (може бути JSON, ідентифікатори тощо).
MaxScore	double	Максимальний бал за питання.
Score	double	Фактично набраний бал користувачем.

3.5. Опис алгоритму перетворення компонентного дерева

Одним із ключових аспектів функціонування системи є можливість зберігати, передавати та відновлювати структуру навчального матеріалу, яка побудована за принципом компонентного дерева. У даному контексті компонентне дерево являє собою ієрархічну модель цифрового навчального контенту, де кожен компонент є окремим логічним елементом (наприклад, текстовий блок, тест, зображення, відео, тощо), який має тип, параметри, вкладені елементи та унікальний ідентифікатор.

Для забезпечення можливості зберігати цю структуру у базі даних або передавати її через мережу, реалізовано алгоритм перетворення дерева у формалізовану структуру, яка підходить для серіалізації у формати JSON або BSON.

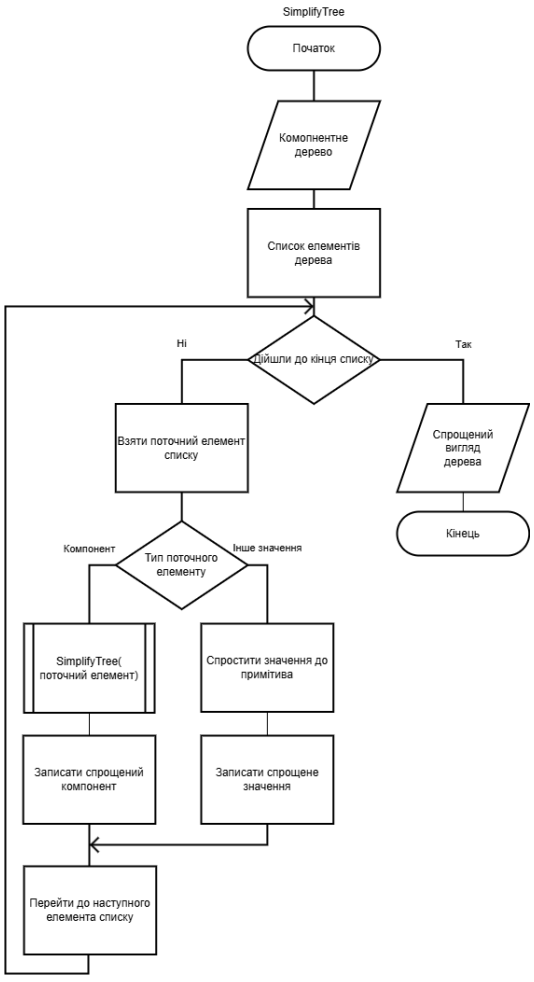


Рис. 3.5. Блок-схема алгоритму перетворення компонентного дерева у загальний вигляд

3.6. Висновки до розділу

У даному розділі було детально розглянуто процес проєктування та розроблення вебзастосунку «StudentPortal», спрямованого на створення та використання цифрових навчальних матеріалів. Сформульовано функціональні та нефункціональні вимоги до програмного забезпечення, що охоплюють основні сценарії взаємодії кінцевих користувачів – студентів і викладачів.

Розглянуто типові сценарії використання системи, які охоплюють створення, редагування, перегляд та публікацію навчального контенту. Описано загальні принципи побудови архітектури програмного забезпечення, зокрема використання зворотного проксі (reverse proxy), модульного підходу та поділу на окремі сервіси, що забезпечують масштабованість і гнучкість системи. Наведено опис моделі даних із деталізацією зв’язків між сутностями.

Окрема увага приділена алгоритмам функціонування системи, зокрема механізму перетворення компонентного дерева освітніх матеріалів у єдине представлення, придатне для збереження у базі даних або передачі мережею.

Результати цього етапу створюють надійну архітектурну та алгоритмічну основу для подальшого розроблення, впровадження, тестування й ефективного використання вебзастосунку в реальному освітньому процесі.

4. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ВЕБЗАСТОСУНКУ

4.1. Особливості реалізації серверної частини та розгортання ПЗ

Серверна частина вебзастосунку є одним із ключових компонентів програмної системи, що забезпечує виконання бізнес-логіки, керування користувачами, оброблення запитів, взаємодію з базами даних та генерацію відповідей клієнтській частині. Надійність, масштабованість та швидкодія серверної логіки безпосередньо впливають на стабільність роботи системи загалом та якість взаємодії з користувачем.

У рамках розроблення вебзастосунку «StudentPortal» для створення та використання цифрових навчальних матеріалів серверна частина була реалізована з використанням платформи ASP.NET Core та мови програмування C#, що дозволяє створювати продуктивні, безпечні й масштабовані вебслужби. Реалізовано RESTful API для взаємодії з клієнтськими інтерфейсами, а також забезпечено підтримку багаторівневої авторизації та автентифікації.

У цьому підрозділі розглядаються архітектурні підходи до побудови серверної частини, вибрані технології, способи оброблення запитів та збереження даних, а також особливості розгортання програмного забезпечення в тестовому та продуктивному середовищах.

4.1.1. Реалізація API та взаємодія з клієнтом

Для забезпечення ефективної взаємодії між клієнтською та серверною частинами у вебзастосунку «StudentPortal» реалізовано архітектуру на основі RESTful API. Такий підхід дозволяє стандартизувати обмін даними, забезпечити розділення відповідальностей між компонентами системи та спростити інтеграцію з різними інтерфейсами, зокрема Angular, Blazor Server та Blazor WebAssembly.

Серверна частина побудована на базі ASP.NET Core, що має вбудовані засоби для створення контролерів, маршрутизації запитів та оброблення

HTTP-відповідей. API реалізовано відповідно до принципів REST, що передбачає використання стандартних методів HTTP (GET, POST, PUT, DELETE) для взаємодії з ресурсами.

Кожен запит, надісланий з клієнта, обробляється відповідним контролером, який доручає логіку оброблення на рівень сервісів. Після оброблення даних формується відповідь у форматі JSON, який легко обробляється у клієнтській-частині. Уся взаємодія між клієнтом і сервером є асинхронною, що підвищує продуктивність та зменшує час очікування користувача.

Особливу роль у цій архітектурі відіграє nginx, який використовується як reverse proxy-сервер. nginx виконує кілька ключових функцій:

1) приймає зовнішні HTTP(S) запити та пересилає їх до серверної частини ASP.NET Core;
2) забезпечує маршрутизацію між кількома сервісами, зокрема frontend-застосунками на Angular і Blazor;
3) кешує статичні ресурси та оптимізує трафік;
4) виконує базові функції балансування навантаження у випадку масштабування;
5) забезпечує закінчення SSL-з’єднань для безпечної передачі даних.

Завдяки застосуванню nginx як проміжного рівня, досягається гнучкість у конфігурації, підвищується безпека, зменшується навантаження на сервер і забезпечується централізований контроль трафіку між клієнтом і сервером. Такий підхід до реалізації API гарантує модульність, спрощує тестування та підтримку застосунку, а також дозволяє легко адаптувати його до нових функціональних вимог у майбутньому.

4.1.2. Робота з базами даних

Ефективна та надійна взаємодія із базами даних є однією з ключових складових функціонування вебзастосунку «StudentPortal». Застосунок обробляє різноманітні типи інформації: від структурованих облікових даних

користувачів до складних ієрархічних навчальних матеріалів, які мають бути швидко доступними й легко редагованими. Для цього використано дві бази даних, кожна з яких виконує окрему роль у загальній архітектурі проєкту.

Основною реляційною системою керування базами даних (СКБД) є PostgreSQL. Вона відповідає за зберігання традиційних структурованих даних: облікові записи користувачів, ролі, інформація про проходження тестів, оцінки, сповіщення та інші службові сутності. PostgreSQL було обрано завдяки її високій продуктивності, надійності та відповідності сучасним вимогам до транзакційності, захисту цілісності даних та широкій підтримці ORM (Object-Relational Mapping) засобів у середовищі ASP.NET Core.

Для роботи з PostgreSQL у проєкті використовується Entity Framework Core, який дозволяє взаємодіяти з базою на рівні об'єктів, забезпечуючи зручність розроблення, підтримку міграцій, перевірку схеми та узгодженість даних. Усі основні запити реалізовано з використанням асинхронних методів, що забезпечує ефективне масштабування та підвищує швидкодію застосунку.

Другою СКБД є MongoDB, яка використовується для зберігання навчальних матеріалів, представлених у вигляді гнучких, деревоподібних документів. Така структура важко реалізується у класичних реляційних базах, але природно і зручно представляється у BSON-документах, які підтримує MongoDB.

Взаємодія з MongoDB здійснюється через офіційний драйвер для C#, що дозволяє працювати з колекціями документів, виконувати фільтрацію, оновлення, агрегації та забезпечує високу продуктивність при обробці великих обсягів матеріалів.

Таким чином, вибір двох баз даних (PostgreSQL і MongoDB), був зумовлений прагненням до ефективного розділення відповідальності: структуровані дані обробляються через реляційну модель, тоді як гнучкі, складні об’єкти зберігаються у документно-орієнтованому середовищі. Це

дозволяє забезпечити як продуктивність, так і масштабованість вебзастосунку, відповідно до сучасних вимог до освітніх платформ.

4.1.3. Система автентифікації та авторизації

У сучасному вебзастосунку важливою складовою безпеки є правильно реалізована система автентифікації та авторизації користувачів. Для забезпечення захищеного доступу до ресурсів і функціональних можливостей застосунку було використано технологію JWT (JSON Web Token).

JSON Web Token – це стандарт токена доступу на основі JSON, стандартизованого в (RFC 7519) [15]. Як правило, використовується для передачі даних для аутентифікації в клієнт-серверних програмах. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який надалі використовує цей токен для підтвердження своєї особи.

У розробленому застосунку автентифікація реалізована за класичною схемою: користувач надає облікові дані (електронну пошту та пароль), які перевіряються на сервері з використанням механізмів ASP.NET Identity. У разі успішної автентифікації користувач отримує JWT-токен, який містить основну інформацію про користувача (ID, ролі, час дії токена тощо) та підписаний за допомогою секретного ключа.

Токен надалі передається клієнтською частиною (Angular або Blazor WASM) в заголовку Authorization при кожному запиті до захищених ресурсів API. Серверна частина, в свою чергу, перевіряє дійсність токена, визначає роль користувача і приймає рішення щодо доступу до відповідної функціональності.

Для реалізації авторизації в застосунку використовуються ролі користувачів, зокрема розмежування доступу між викладачами та студентами. Відповідні політики доступу налаштовуються в конфігурації ASP.NET Core через AuthorizationPolicy.

Такий підхід дозволяє створити гнучку та масштабовану систему

безпеки, яка легко інтегрується з будь-яким типом клієнтського інтерфейсу і забезпечує надійний захист персональних даних користувачів та логіки застосунку.

4.1.4. Розгортання застосунку

Процес розгортання вебзастосунку є завершальним етапом у циклі його розроблення, що забезпечує доступність системи для кінцевих користувачів. У розробленому рішенні особливу увагу приділено стабільності, масштабованості та зручності обслуговування при розгортанні.

Архітектура застосунку побудована за принципами розділення обов'язків: окремо функціонують серверна частина, клієнтська частина (Angular, Blazor) та служби баз даних (PostgreSQL, MongoDB). Для забезпечення ефективної взаємодії між компонентами використовується nginx як reverse proxy-сервер, що виконує роль маршрутизатора HTTP-запитів до відповідних сервісів, а також забезпечує базовий рівень захисту, кешування та управління статичними ресурсами.

Розгортання відбувається на основі контейнеризації, використовуючи Docker. Кожен компонент (API, клієнтський інтерфейс, бази даних) упаковано в окремий контейнер, що дозволяє швидко запускати, оновлювати або масштабувати систему без ризику конфліктів між середовищами. Для координації роботи контейнерів застосовується Docker Compose, що дозволяє описати всю інфраструктуру застосунку у вигляді конфігураційного файлу та запускати її єдиною командою.

Безпосередньо перед розгортанням виконується процес збірки, в якому:

1) Angular-компонент компілюється до статичних файлів;

2) Blazor WASM будується та копіюється у відповідний каталог.

Після збору всі сервіси розгортаються на сервері (локальному чи хмарному), а nginx перенаправляє запити на відповідні порти. Бази даних

запускаються у своїх контейнерах, з попередньо налаштованими змінними середовища та правилами зберігання даних.

У якості середовища для розгортання може використовуватись будь-який сучасний хостинг або VPS, що підтримує Docker.

Такий підхід до розгортання забезпечує:

1) простоту підтримки та оновлення;

2) гнучкість масштабування;

3) легке дублювання середовища для тестування або розроблення;

4) підвищену безпеку за внаслідок ізоляції компонентів.

4.1.5. Забезпечення продуктивності та безпеки

Під час розроблення вебзастосунку особливу увагу було приділено двом ключовим аспектам: продуктивності та інформаційній безпеці. Обидва фактори безпосередньо впливають на стабільність, надійність і довіру користувачів до системи, особливо в контексті освітнього середовища, де обробляються особисті дані студентів і викладачів, а також результати тестування.

Продуктивність

Для забезпечення високої продуктивності застосовано такі підходи:

1. Розділення навантаження між клієнтською частиною та сервером. Обчислювально затратні операції, зокрема генерація даних або оброблення результатів тестування, виконуються на сервері, тоді як інтерфейс користувача працює максимально незалежно через Blazor WASM.
2. Кешування на рівні nginx та в самому застосунку. Статичні файли (Angular-інтерфейс, бібліотеки) кешуються браузером, що мінімізує час завантаження сторінок.
3. Оптимізовані запити до бази даних. Використовуються проєкції, індексація в PostgreSQL та агрегації в MongoDB для зменшення обсягу переданих даних.

4. Асинхронне програмування. Усі запити до бази даних і мережеві операції реалізовані з використанням асинхронних методів, що підвищує здатність серверу обробляти паралельні запити.

Безпека

З метою гарантування безпечної взаємодії користувача з системою впроваджено такі заходи:

1. JWT (JSON Web Tokens) використовується для автентифікації. Кожен користувач отримує токен з обмеженим терміном дії, який передається в заголовках HTTP-запитів. Сервер перевіряє справжність токена, а також ролі користувача.
2. Паролі користувачів зберігаються у вигляді хешу, використовуючи механізми ASP.NET Identity з сучасними алгоритмами хешування (наприклад, PBKDF2).
3. Захист від CSRF-атак забезпечено завдяки стандартним middleware у Blazor Server та Angular, а також правильній конфігурації cookie;
4. Валідація даних виконується як на клієнтській, так і на серверній стороні. Це допомагає запобігти ін’єкціям(коду, стилів, SQL), некоректним запитам і потенційно шкідливим діям користувачів.
5. Розмежування доступу реалізовано через ролі та політики авторизації. Кожен запит перевіряється на предмет прав доступу до відповідних ресурсів, що унеможливлює несанкціонований доступ до даних.
6. Регулярне логування та аудит подій входу, спроб несанкціонованого доступу, змін у базі даних тощо, дозволяє моніторити стан системи й виявляти потенційні загрози.
7. HTTPS обов’язково використовується при розгортанні, що захищає всі передані дані від перехоплення або модифікації.

Реалізація описаних підходів дозволяє забезпечити надійну роботу системи навіть при значному навантаженні, а також гарантувати безпеку персональних даних і результатів навчання користувачів.

4.2. Опис візуальної компоненти застосунку

Візуальна компонента застосунку відіграє ключову роль у забезпеченні зручного та інтуїтивно зрозумілого користувацького досвіду. Основна мета цієї частини – надати користувачу зрозумілий інтерфейс для виконання базових операцій: створення, редагування та перегляд навчальних матеріалів, проходження тестів, а також моніторингу результатів.

Інтерфейс побудований із використанням сучасних клієнт-технологій, які забезпечують високу швидкодію, адаптивність та інтерактивність. Для реалізації застосовано компоненти, що відповідають принципам модульності та повторного використання, що сприяє спрощенню подальшої підтримки та масштабування.

Основні елементи інтерфейсу включають:

1) головна сторінка: містить вхід до системи та короткий опис можливостей застосунку;
2) сторінка авторизації та реєстрації: забезпечує безпечний доступ користувачів із можливістю перемикання між формами;
3) панель користувача: відображає особисті дані, список створених матеріалів, результати тестів, а також надає швидкий доступ до основних функцій;
4) редактор матеріалів: інтерактивний інструмент зі зручним вибором шаблонів, можливістю копіювати та вставляти компоненти, а також автозбереженням прогресу;
5) сторінка тестування: забезпечує інтуїтивний процес проходження тесту з автоматичним збереженням результатів у реальному часі;
6) сторінка оцінювання: викладач може легко переглядати деталі відповідей студентів, сортувати і шукати за іменами чи статусами тестів.

Колірна палітра, типографіка та розташування елементів було розроблено відповідно до вимог контрастності та зручності сприйняття

інформації, що сприяє доступності інтерфейсу для користувачів із різними потребами.

Таким чином, візуальна компонента застосунку забезпечує злагоджену взаємодію користувача з системою, підтримуючи ефективність та комфорт роботи на всіх рівнях.

4.2.1. Опис сторінок

Головна сторінка (рис. 4.1) застосунку виконує роль стартової точки для користувачів, які тільки знайомляться з застосунком. Основними її функціональними можливостями є:

Навігація

Перехід до розділу "FAQ": забезпечує доступ до інформації щодо функціональності застосунку, відповідей на поширені запитання. Кнопка "Log In": веде на сторінку авторизації користувача для входу в систему. Кнопка "Join Now": відкриває форму реєстрації нового користувача.

Інформаційна презентація

Заголовок та опис платформи: короткий виклад мети та переваг використання системи, орієнтований на освітян. Мотиваційний текст: закликає до дії, підкреслює зручність використання, економію часу та ефективність платформи.

Інформаційні блоки з поясненням функціональності

У розділі "How it works" представлено три інтерактивні блоки:

1. Створення матеріалів: пояснює, що користувачі можуть швидко створювати, редагувати та повторно використовувати навчальні матеріали.
2. Збереження ресурсів: демонструє можливість організованого зберігання уроків, тестів і матеріалів у межах платформи.
3. Автоматизація оцінювання: інформує про підтримку автоматичного оцінювання, надання зворотного зв’язку та відстеження прогресу учнів.

Дизайн та доступність

Адаптивний дизайн: сторінка оптимізована для коректного відображення на різних пристроях (персональні комп'ютери, планшет, мобільні телефони). Контрастність та зручність читання: використано чітку типографіку, візуальні акценти та логічну структуру елементів.

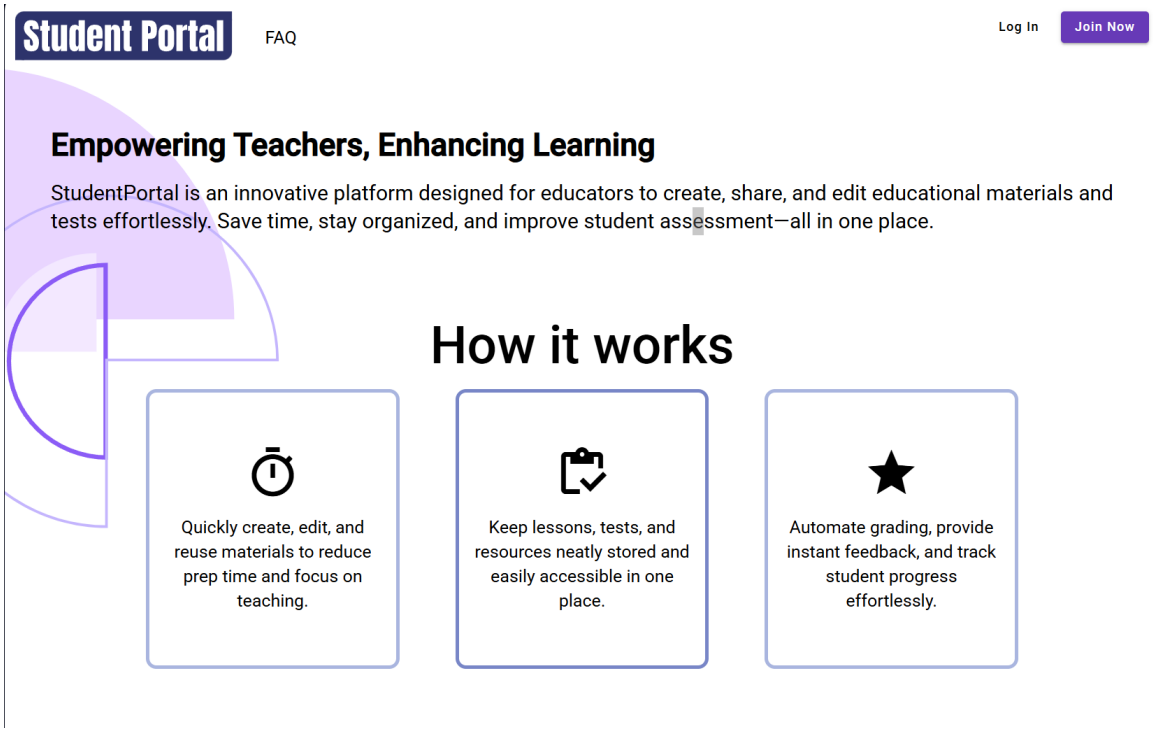


Рис. 4.1. Головна сторінка

Сторінка реєстрації (рис. 4.2) дозволяє новим користувачам створити обліковий запис у системі, обравши одну з ролей (наприклад, учень або вчитель). Вона забезпечує необхідну валідацію введених даних і є невід'ємною частиною системи автентифікації.

Форма реєстрації має обов'язкові поля з такою валідацією:

- email повинен бути у форматі email-адреси;
- паролі мають збігатися;
- роль повинна бути обрана перед надсиланням форми.

Кнопка Submit

Submit – після натискання, дані перевіряються на відповідність

умовам, і відбувається надсилання запиту на створення нового акаунта. При успішній реєстрації користувач автоматично перенаправляється до внутрішньої частини системи або сторінки входу.

Альтернативне посилання

"Already have an account? Sign in" – текстове посилання під формою, яке перенаправляє на сторінку авторизації, якщо користувач уже має акаунт. UI/UX Особливості Захищене введення пароля з можливістю показати/сховати введене значення.

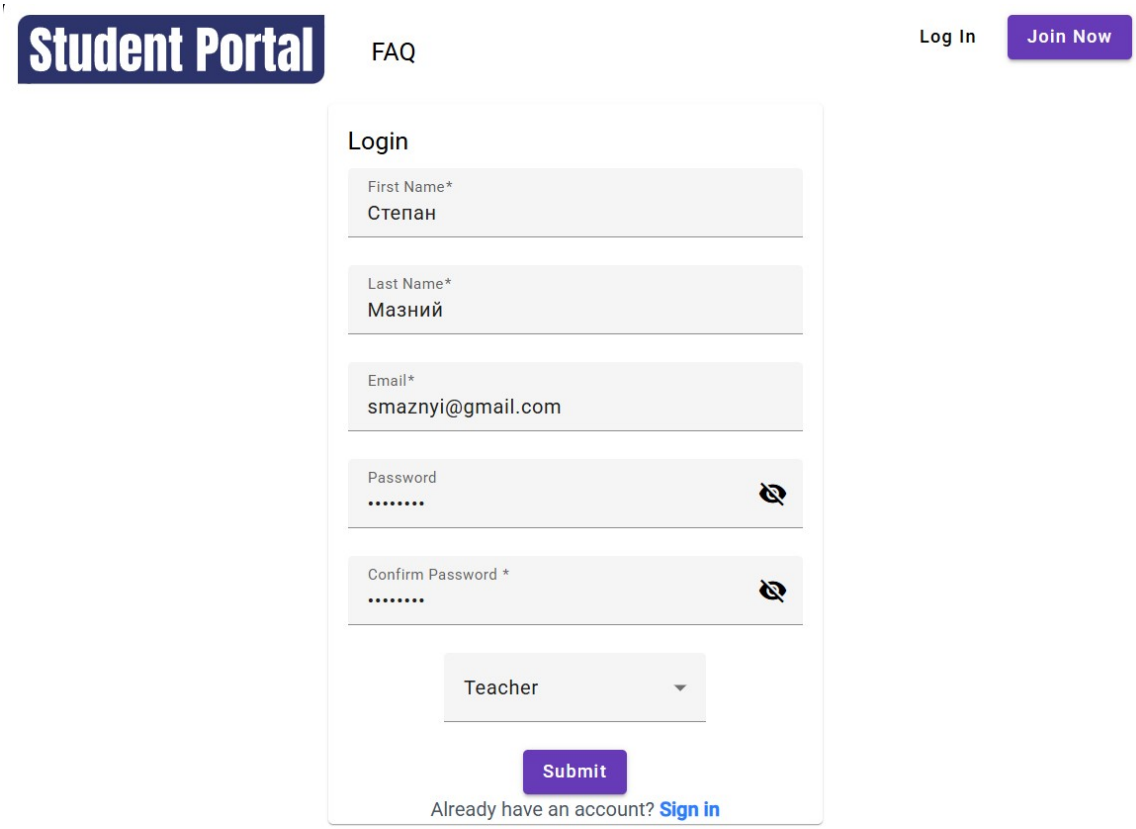


Рис. 4.2. Сторінка реєстрації

Сторінка авторизації (рис. 4.3) дозволяє зареєстрованим користувачам входити до системи «StudentPortal», використовуючи email та пароль. Забезпечує перевірку введених даних та надає зворотний зв’язок у разі помилок.

Форма авторизації

Складається з двох основних полів:

- email – поле введення електронної пошти користувача;
- password – поле введення пароля (із можливістю відобразити/сховати символи).

Повідомлення про помилку

Якщо введені дані некоректні (наприклад, неправильний пароль або email не знайдено), під кнопкою “Submit” з’являється повідомлення про помилку.

Кнопка Submit

Submit – ініціює перевірку даних на сервері. Якщо дані правильні – користувач авторизується і потрапляє до основної частини платформи. Кнопка неактивна до моменту заповнення обох полів.

Альтернативне посилання

“Don’t have an account? Sign up” – перенаправляє нового користувача на сторінку реєстрації.

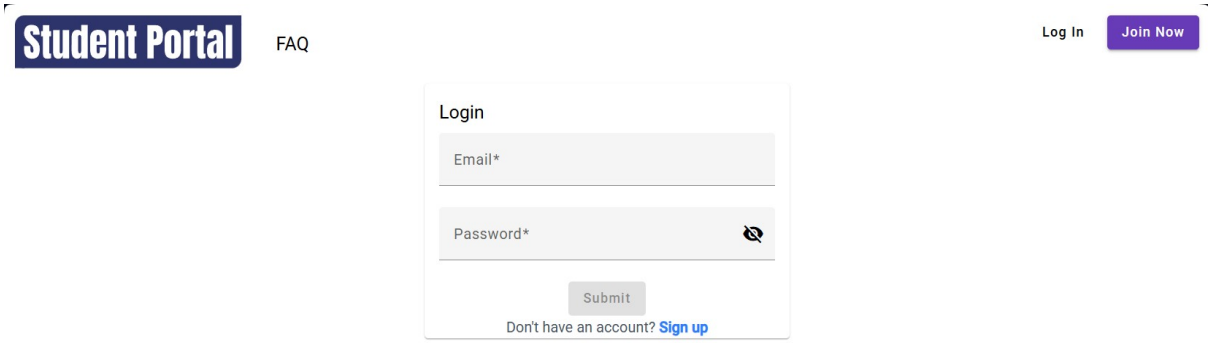


Рис. 4.3. Сторінка входу в акаунт

Сторінка редагування даних акаунту (рис. 4.4) дозволяє зареєстрованому користувачеві переглядати та редагувати особисту інформацію в межах «StudentPortal». Вона надає можливість оновити ім'я, прізвище, електронну пошту та роль, а також завантажити або змінити

аватар і змінити пароль.

Кнопка "Submit" (Підтвердити)

Submit – ініціює процес збереження внесених змін до профілю користувача. Кнопка стає активною після внесення змін у будь-яке з полів форми.

Завантаження аватара

Блок з відображенням поточного аватара користувача (за наявності). Кнопка "Upload avatar" (Завантажити аватар) – відкриває діалогове вікно для вибору файлу зображення з пристрою користувача.

Зміна пароля

Кнопка "Change password" (Змінити пароль) – перенаправляє користувача на окрему сторінку або відкриває модальне вікно для зміни пароля.

Рис. 4.4. Сторінка редагування даних акаунту

Сторінка матеріалів (рис. 4.5) надає користувачу інтерфейс для управління матеріалами, які представлені у вигляді карток. Кожна картка відображає назву матеріалу та містить елементи керування для взаємодії з

ним.

Меню дій (три крапки)

У правому верхньому куті кожної картки матеріалу розташована іконка з трьома крапками, що відкриває контекстне меню при натисканні.

Контекстне меню містить наступні дії:

- Open (Відкрити) – відкриває матеріал для перегляду або редагування (дублює дію при натисканні на саму картку).
- Rename (Перейменувати) – відкриває діалогове вікно для введення нової назви матеріалу.
- Delete (Видалити) – ініціює процес видалення матеріалу (може вимагати підтвердження).
- Clone (Клонувати) – створює копію поточного матеріалу з новою назвою (зазвичай додається "(копія)" або подібне).

Додавання нового матеріалу

У правому верхньому куті сторінки розташована кнопка з іконкою "+" (плюс). При натисканні на цю кнопку відкривається діалогове вікно з можливістю вибору шаблону для нового матеріалу. Після вибору шаблону створюється новий матеріал, який відображається на сторінці у вигляді нової картки (зазвичай з назвою за замовчуванням, яку можна змінити).

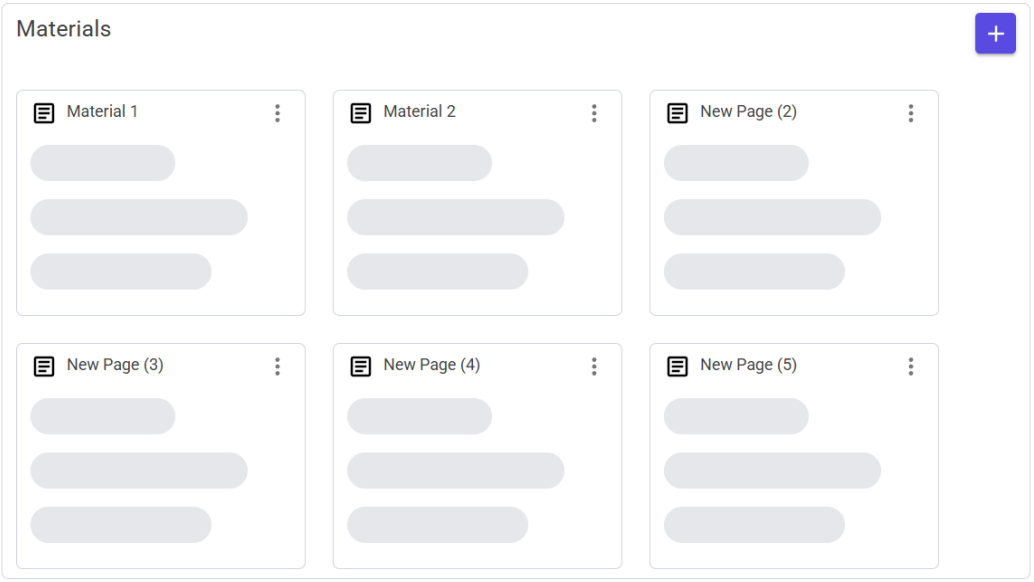


Рис. 4.5. Сторінка матеріалів вчителя

Сторінка редагування матеріалів (рис. 4.6), призначена для створення та модифікації навчальних матеріалів. Вона надає користувачеві інструменти для форматування тексту, додавання мультимедійного контенту та попереднього перегляду результату. В залежності від обраного шаблону при створенні матеріалу може мати різні можливості.

Заголовок сторінки та керування назвою

У верхній лівій частині сторінки розташоване поле Page Name (Назва сторінки), яке дозволяє користувачеві задати або змінити назву поточного матеріалу. Ліворуч від назви сторінки розташовані кнопки контролю історії дій:

- Стрілка назад – відповідає за скасування останньої дії, аналогічно комбінації клавіш Ctrl+Z.
- Стрілка вперед – відповідає за повернення скасованої дії, аналогічно комбінації клавіш Ctrl+Shift+Z.

Перемикач режиму відображення

Праворуч знаходиться випадаючий список з назвою, що відображає поточний режим відображення вікна редактора.

Кнопки дій

Праворуч від перемикача режиму розташовані дві основні кнопки:

- SHARE (Поділитися) – при натисканні відкриває діалогове вікно з посиланням для перегляду матеріалу іншими користувачами.
- SAVE (Зберегти) – ініціює процес збереження поточних змін, внесених до матеріалу.

Область редактора матеріалів

Основну частину сторінки займає область редактора, яка складається з інструментів форматування та області введення і редагування контенту матеріалу.

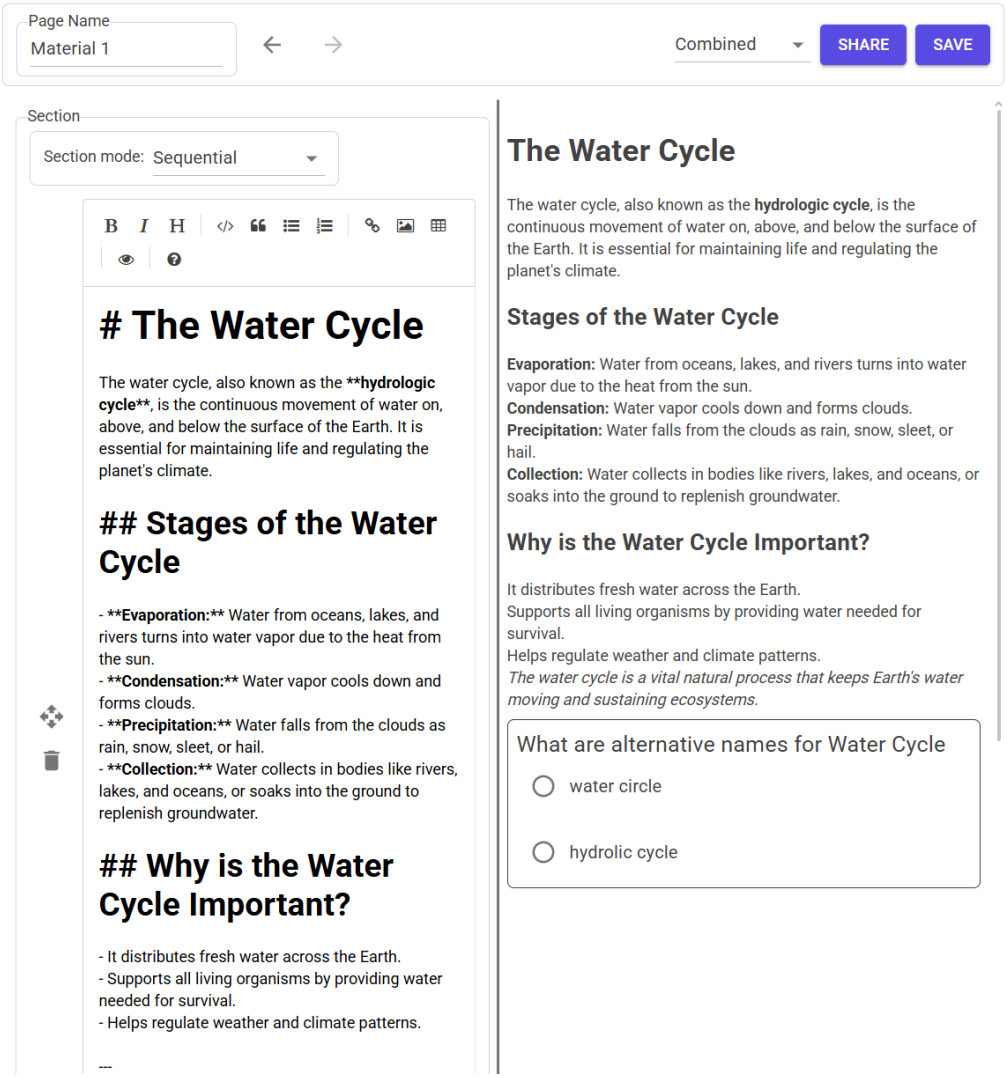


Рис. 4.6. Сторінка редактора

4.3. Особливості проведеного тестування

Тестування застосунку «StudentPortal» проводилось на кількох рівнях, з метою забезпечення стабільності, надійності та відповідності функціональним вимогам. Було реалізовано модульне тестування бізнес-логіки, перевірку API- інтерфейсів та тестування коректності відображення і поведінки інтерфейсу.

Unit-тестування (модульне тестування)

Unit-тести охоплювали ключові модулі застосунку, що не пов’язані з інтерфейсом, зокрема:

1) обробка логіки реєстрації та авторизації користувачів – перевірка правильності валідації даних, хешування паролів, генерації токенів

доступу;

2) управління навчальними матеріалами – створення, редагування, збереження та повторне використання ресурсів і тестів;
3) логіка перевірки відповідей – алгоритми автоматичної оцінки, зіставлення правильних відповідей, нарахування балів;
4) системи ролей і доступу – перевірка розмежування прав доступу між учнями та вчителями.

Тестування API

REST API проходило окреме тестування з метою перевірки:

1) коректності відповідей (формат, статус-коди, вміст);
2) оброблення некоректних запитів (відсутні параметри, неправильні типи даних, несанкціонований доступ);
3) стійкості до навантаження – для критичних маршрутів (наприклад, логін/тестування/створення тестів);
4) захищеності даних – автентифікація, авторизація, токени доступу, обмеження за ролями;
5) API тестувалося за допомогою Postman.

Тестування інтерфейсу користувача

Коректність роботи інтерфейсу перевірялась як вручну, так і частково автоматизовано, з акцентом на:

1) візуальну відповідність макетам;
2) правильність навігації між сторінками (реєстрація, логін, створення тесту, перегляд результатів тощо);
3) валідацію форм – перевірка повідомлень про помилки, некоректний формат email, порожні поля;
4) повідомлення про успішні/неуспішні дії – наприклад, при неправильному логіні або створенні тесту;
5) реакцію на дії користувача – інтерактивні елементи, кнопки, спливаючі повідомлення тощо;
6) інтерфейс тестувався у популярних браузерах (Chrome, Firefox,

Edge) для перевірки кросбраузерної сумісності та коректного відображення.

Цей підхід до тестування дозволив виявити й усунути помилки на ранніх етапах розроблення, підвищити надійність коду та забезпечити якісний досвід взаємодії для кінцевих користувачів.

4.4. Рекомендації щодо подальшого вдосконалення

У процесі розробки вебзастосунку було реалізовано базову функціональність, необхідну для створення, редагування та перегляду цифрових навчальних матеріалів. Водночас, з урахуванням перспектив розвитку системи та потреб освітнього процесу, доцільно передбачити такі напрямки подальшого вдосконалення:

- Інтеграція з LMS-системами – забезпечить синхронізацію курсів, прогресу студентів та оцінювання з існуючими платформами (наприклад, Moodle або Google Classroom).
- Розширення типів завдань – додавання відкритих запитань, інтерактивних вправ, підтримки завантаження файлів користувачами.
- Підключення аналітики навчального процесу – впровадження модулів для аналізу успішності студентів, відстеження активності та побудови персоналізованих рекомендацій.
- Підтримка офлайн-доступу – реалізація механізмів для роботи з матеріалами без постійного з'єднання з мережею, з подальшою синхронізацією.
- Розширення мовної підтримки – інтерфейсна локалізація для забезпечення доступності платформи для ширшої аудиторії.

Зазначені рекомендації дозволять не лише підвищити функціональність системи, а й зробити її гнучким, масштабованим інструментом для повноцінної підтримки сучасного освітнього процесу.

4.5. Висновки до розділу

У результаті реалізації та тестування застосунку «StudentPortal» було досягнуто високого рівня функціональності, стабільності та зручності у користуванні. Проведене модульне тестування підтвердило коректну роботу основних логічних компонентів, зокрема механізмів реєстрації, авторизації, створення та оцінювання тестів.

Тестування API забезпечило впевненість у надійності серверної-частини системи, включаючи оброблення запитів, авторизацію та обмін даними. Перевірка інтерфейсу засвідчила, що застосунок є інтуїтивно зрозумілою, адаптивною та зручною як для викладачів, так і для студентів.

Таким чином, розроблений застосунок відповідає поставленим вимогам, є готовим до використання в реальних умовах навчального процесу та має потенціал для подальшого масштабування й вдосконалення.

У рамках аналізу перспектив розвитку системи було визначено низку напрямів, які дозволять суттєво розширити функціональність і підвищити ефективність її використання. Зокрема, рекомендовано реалізувати інтеграцію з існуючими LMS, додати модулі аналітики навчальної діяльності, запровадити механізми адаптивного навчання та підтримку офлайн-доступу.

ВИСНОВКИ

У процесі виконання дипломного проєкту було розроблено вебзастосунок «StudentPortal», який реалізує функціональність для створення, редагування, поширення та перевірки навчальних матеріалів і тестових завдань. Система орієнтована на задоволення потреб викладачів і студентів у контексті сучасного цифрового освітнього середовища.

На основі аналізу предметної області та існуючих рішень було сформовано вимоги до функціональності застосунку, спроєктовано архітектуру програмного продукту та реалізовано користувацький інтерфейс і серверну логіку. Особливу увагу приділено зручності користування, безпеці автентифікації, а також можливості масштабування проєкту в майбутньому.

У якості основних засобів реалізації було використано технології ASP.NET Core для побудови серверної частини, Angular для створення інтерактивного клієнтського інтерфейсу, а також Blazor Server і Blazor WebAssembly – для забезпечення швидкого рендерингу контенту та редагування навчальних матеріалів у реальному часі. Для зберігання даних застосовано PostgreSQL (структуровані дані) та MongoDB (неструктуровані дані). Аутентифікація реалізована на основі JSON Web Token (JWT), що гарантує безпечну та масштабовану модель доступу. Reverse proxy-сервер на базі Nginx забезпечує маршрутизацію запитів і балансування навантаження.

Проведене тестування підтвердило стабільну та коректну роботу основних компонентів системи. Було реалізовано модульні тести для логіки застосунку, перевірено роботу API та здійснено тестування інтерфейсу для виявлення потенційних помилок та покращення досвіду використання.

Розроблений програмний продукт відповідає сучасним вимогам до вебзастосунків освітнього призначення, може бути використаний у закладах освіти, а також має перспективи для подальшого розширення функціональнсоті – зокрема, шляхом інтеграції з LMS-системами,

додаванням аналітики та адаптивного навчання.

Результати дипломного проєкту засвідчують доцільність та ефективність створення подібного інструменту для автоматизації та підтримки освітнього процесу.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Цифрові навчальні матеріали. [Електронний ресурс]. – Режим доступу: https://nuschool.eu/lessons/elementary/information/22.html. – Дата доступу: 21.05.2025 р.
2. Google Classroom. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Google_Classroom. – Дата доступу: 21.05.2025 р.
3. Moodle. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Moodle. – Дата доступу: 21.05.2025 р.
4. Canva. [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Canva. – Дата доступу: 21.05.2025 р.
5. Node.js. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Node.js. – Дата доступу: 21.05.2025 р.
6. Django. [Електронний ресурс]. – Режим доступу: https://www.djangoproject.com. – Дата доступу: 21.05.2025 р.
7. Spring Boot. [Електронний ресурс]. – Режим доступу: https://spring.io/projects/spring-boot. – Дата доступу: 21.05.2025 р.
8. ASP.NET Core. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-9.0. – Дата доступу: 21.05.2025 р.
9. React. [Електронний ресурс]. – Режим доступу: https://react.dev. – Дата доступу: 21.05.2025 р.
10. Vue.js. [Електронний ресурс]. – Режим доступу: https://vuejs.org. – Дата доступу: 21.05.2025 р.
11. Angular. [Електронний ресурс]. – Режим доступу: https://angular.dev. – Дата доступу: 21.05.2025 р.
12. Blazor. [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Blazor. – Дата доступу: 21.05.2025 р.
13. PostgreSQL. [Електронний ресурс]. – Режим доступу:

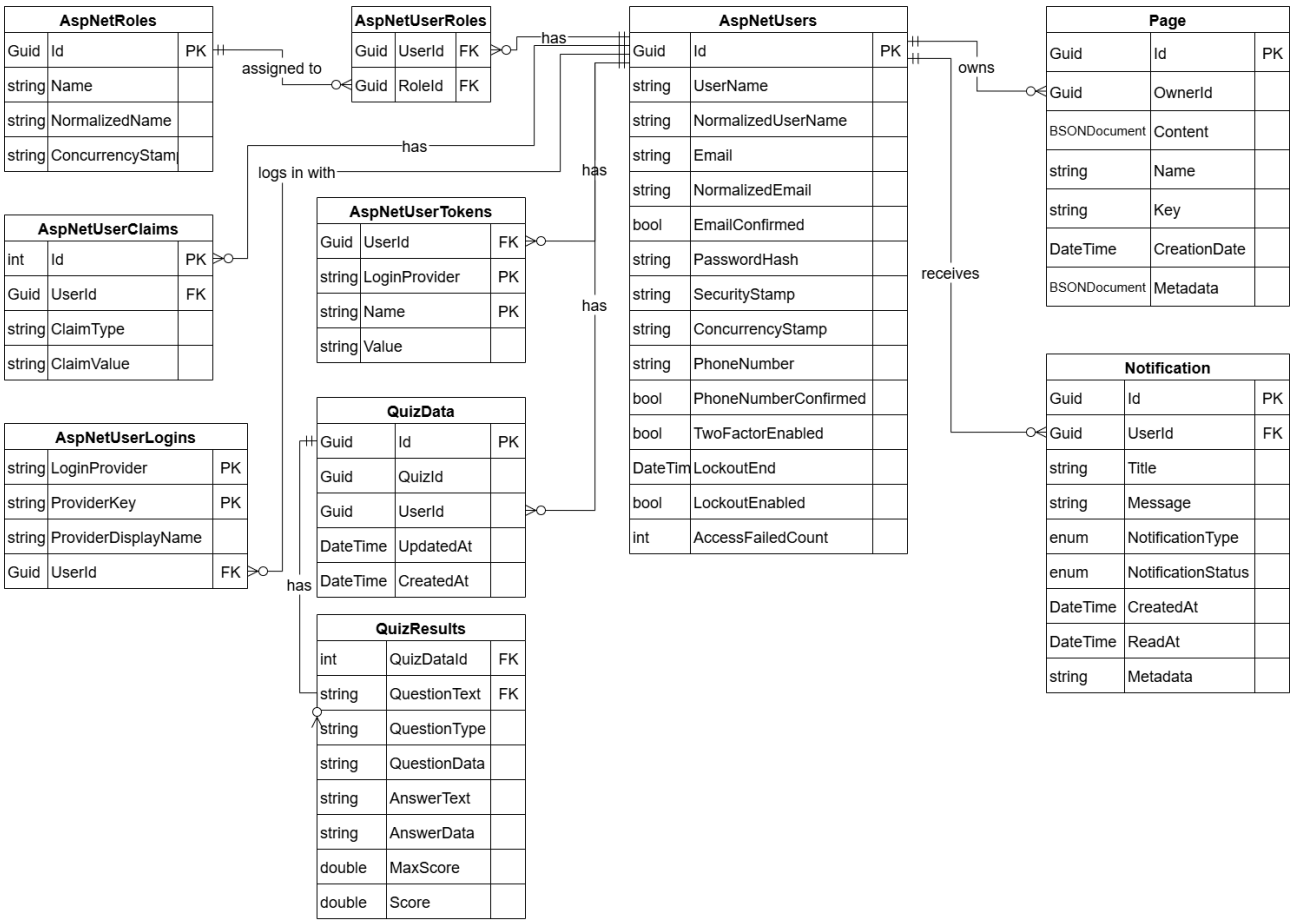
https://www.postgresql.org. – Дата доступу: 21.05.2025 р.

14. MongoDB. [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/. – Дата доступу: 21.05.2025 р.
15. JSON Web Token. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/JSON_Web_Token. – Дата доступу: 21.05.2025 р.

ДОДАТКИ

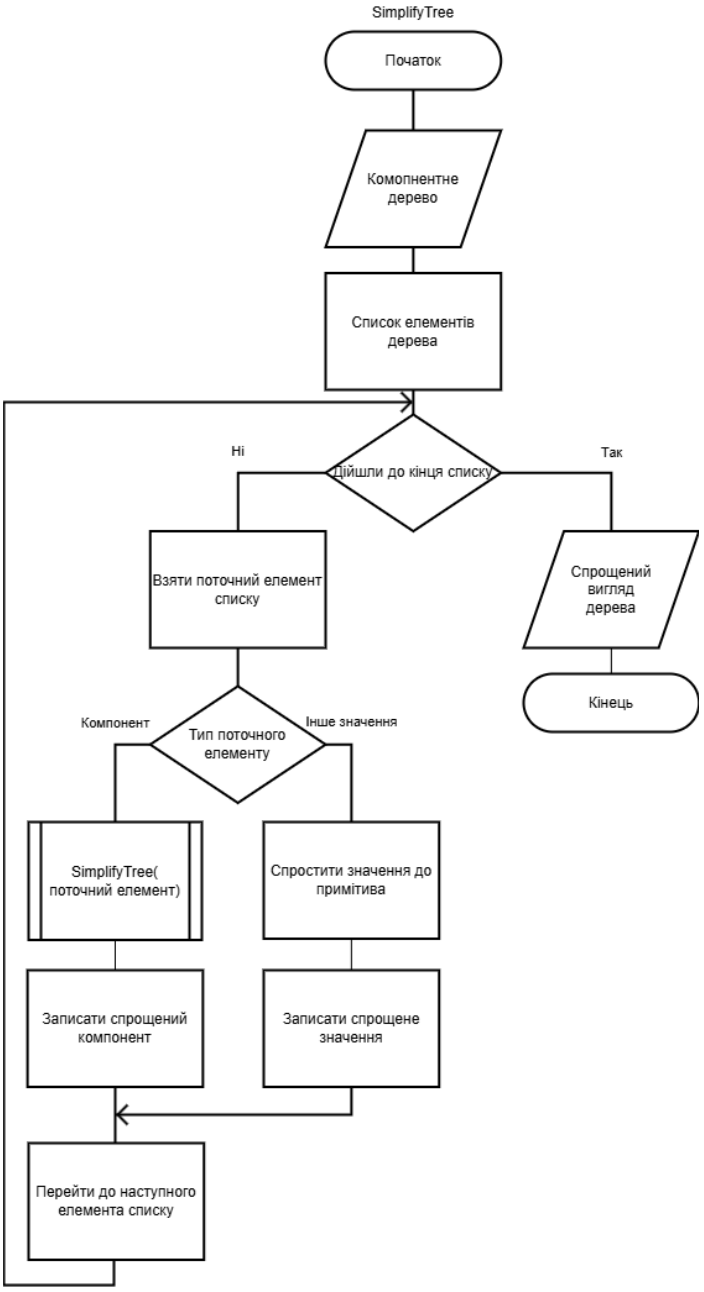
Додаток 1

Копії графічних матеріалів



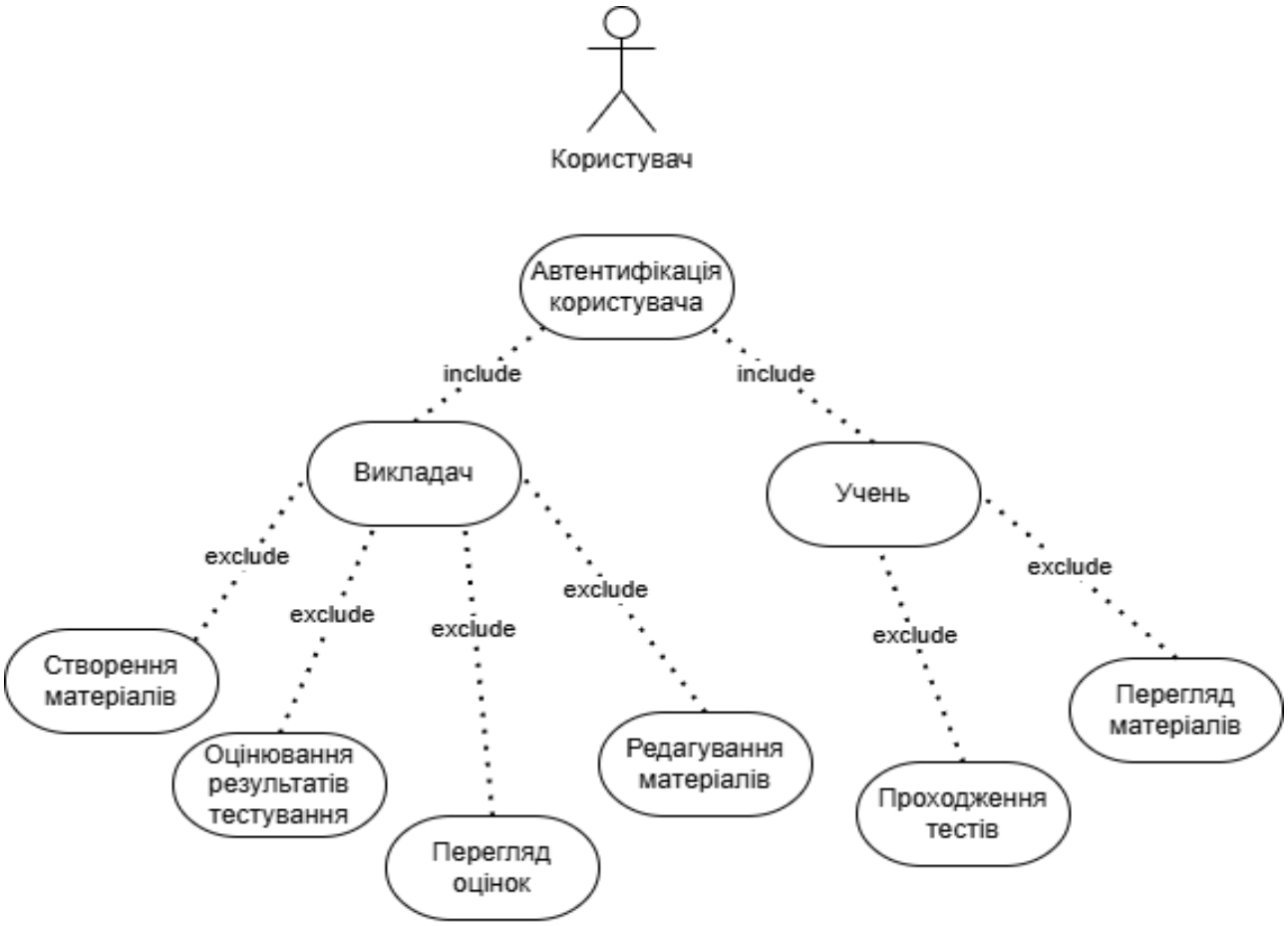
ДП.045440-06-99

Вебзастосунок для створення та використання цифрових навчальних матеріалів. Діаграма взаємозв’язків між сутностями. ER-діаграма



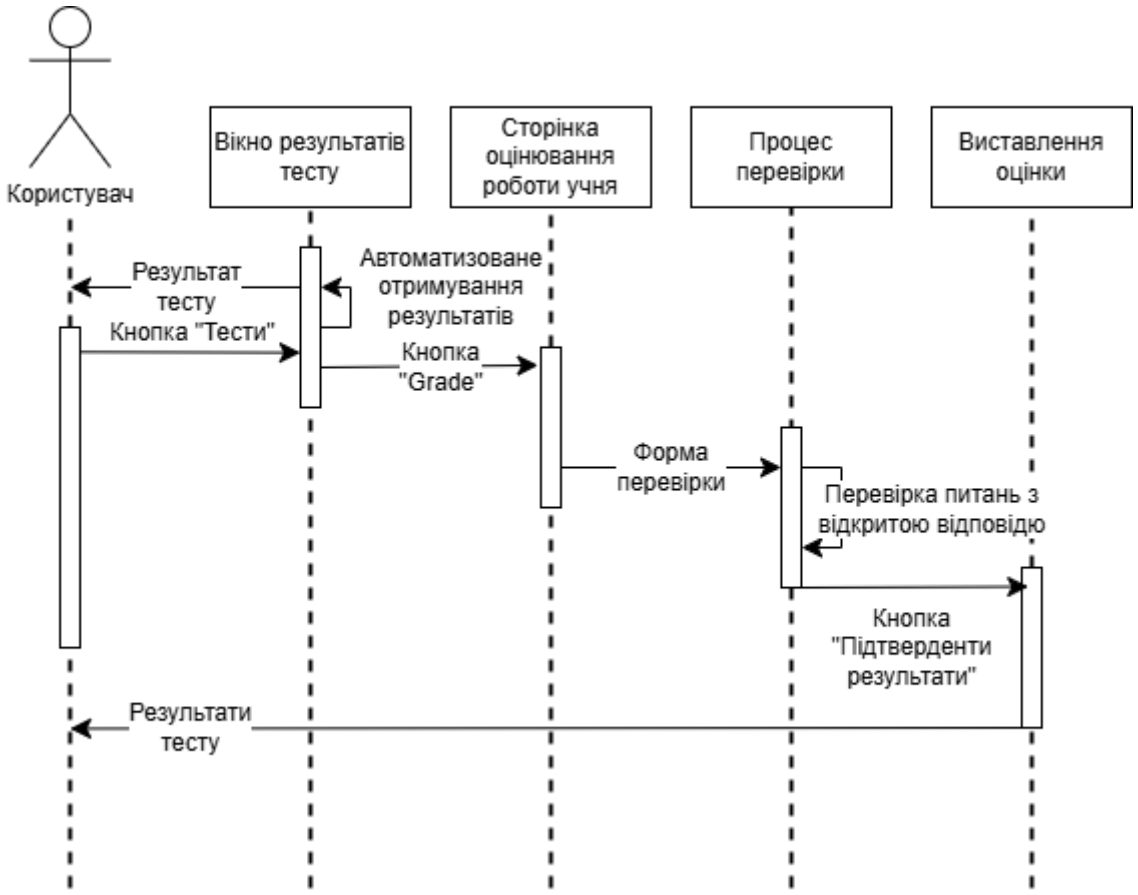
ДП.045440-07-99

Вебзастосунок для створення та використання цифрових навчальних матеріалів. Алгоритм перетворення компонентного дерева у загальний вигляд. Блок-схема алгоритму



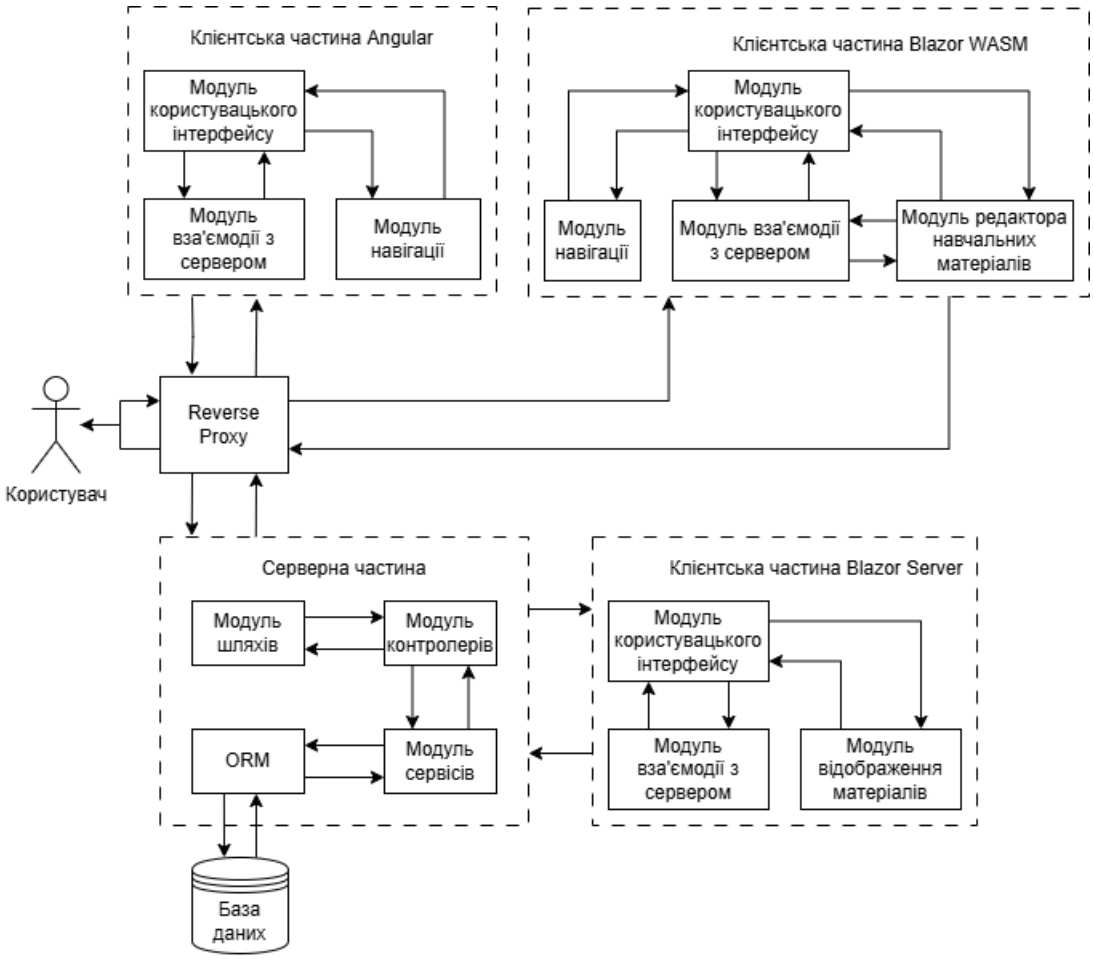
Діаграма прецедентів розробленого застосунку

Мазний С. В., група КП-11



Граф-модель сценарію «Отримання результатів тесту викладачем»

Мазний С. В., група КП-11



Узагальнена архітектура вебзастосунку

Мазний С. В., група КП-11

Додаток 2

Лістинг програми

Фрагмент коду авторизації

```
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using System.Text.Json;
using FluentValidation;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;
using Microsoft.IdentityModel.Tokens;
using StudentPortal.AuthService.Entities;

namespace StudentPortal.AuthService.Controllers;

[ApiController, Route("/")]
public class AuthController(
    UserManager<ApplicationUser> userManager,
    SignInManager<ApplicationUser> signInManager,
    IOptions<JWTConfig> jwtConfig,
    ILogger<AuthController> logger) : ControllerBase
{
    [HttpPost("register")]
    public async Task<ActionResult<UserDTO>> Register([FromBody] RegisterDTO
model)
    {
        logger.LogInformation("Trying to register new user: {Email}",
model.Email);
        var user = new ApplicationUser()
        {
            UserName = model.Email,
            FirstName = model.FirstName,
            LastName = model.LastName,
            Email = model.Email,
        };

        IdentityResult result;
        try
        {
            result = await userManager.CreateAsync(user, model.Password);
        }
        catch
        {
            logger.LogError("Error registering a user {Email}", user.Email);
            return BadRequest();
        }

        if (result.Succeeded)
        {
            await userManager.AddToRoleAsync(user, model.Role);

            logger.LogInformation("A new user has been registered: Username:
{Username}, FirstName: {FirstName}, LastName: {LastName}, Email: {Email}",
                user.UserName,
                user.FirstName,
                user.LastName,
                user.Email);

            IList<string> roles = [model.Role];

            var token = IssueToken(user, roles);
```

```
            SetAuthCookie(token);

            return Ok(new UserDTO()
            {
                Id = user.Id,
                FirstName = user.FirstName,
                LastName = user.LastName,
                Email = user.Email,
                Roles = roles,
                JwtToken = token
            });
        }

        logger.LogInformation("Could not register new user. Username:
{Username}, FirstName {FirstName}, LastName: {LastName}. Errors: {Errors}",
user.UserName, user.FirstName, user.LastName,
JsonSerializer.Serialize(result.Errors));

        return BadRequest(result.Errors);
    }

    [HttpPost("login")]
    public async Task<ActionResult<UserDTO>> Login([FromBody] LoginDTO model)
    {
        var result = await signInManager.PasswordSignInAsync(model.Email,
model.Password, false, false);

        if (!result.Succeeded)
        {
            logger.LogInformation("User login failed. User: {Username}",
model.Email);
            return Unauthorized();
        }

        var user = (await userManager.FindByNameAsync(model.Email))!;

        logger.LogInformation("User login successful. User: {Username}",
model.Email);

        var roles = await userManager.GetRolesAsync(user);

        var token = IssueToken(user, roles);

        SetAuthCookie(token);

        return Ok(new UserDTO()
        {
            Id = user.Id,
            FirstName = user.FirstName,
            LastName = user.LastName,
            Email = user.Email!,
            Roles = roles,
            JwtToken = token
        });
    }

    private void SetAuthCookie(string token)
    {
        HttpContext.Response.Cookies.Append("AuthToken", token, new
CookieOptions() { HttpOnly = true, Secure = true, SameSite = SameSiteMode.None,
Expires = DateTimeOffset.UtcNow.AddMonths(1) });
    }
```

```
    private string IssueToken(ApplicationUser user, IList<string> roles)
    {
        var securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtConfig.Value.SecretKey));
        var credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);

        List<Claim> claims = [
            new Claim(JwtRegisteredClaimNames.Sub, user.Id),
            new Claim(ClaimTypes.NameIdentifier, user.UserName!),
            new Claim(ClaimTypes.Email, user.Email!)
        ];

        foreach (var role in roles)
        {
            claims.Add(new Claim(ClaimTypes.Role, role));
        }

        var token = new JwtSecurityToken(
            issuer: jwtConfig.Value.Issuer,
            audience: jwtConfig.Value.Audience,
            claims,
            expires: DateTime.Now.AddMonths(1),
            signingCredentials: credentials
        );

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
```

Фрагмент коду сторінки редактора

```
@page "/{id}"
@using PageEditor.Components.Editor
@using StudentPortal.CommonComponents
@using StudentPortal.CommonComponents.Renderer
@using StudentPortal.ComponentData.Abstractions
@using StudentPortal.ComponentData.Components
@using StudentPortal.ComponentData.Conversion
@using StudentPortal.PageEditor.Components
@using System.Text.Json
@using StudentPortal.PageEditor.Storage
@using StudentPortal.PageEditor.Templates
@using StudentPortal.Services
@using System.Collections.Immutable

@inherits FluxorComponent
@inject IState<EditorState> EditorState
@inject IDispatcher Dispatcher
@inject NavigationManager NavigationManager
@inject IQuizManager QuizManager

@code
{
    [Parameter]
    public required string Id { get; set; }

    private IPageTemplate GetTemplate()
    {
        var templateName = EditorState.Value.Page.Metadata["Template"];
```

```
        switch (templateName)
        {
            case "quiz": return new QuizPageTemplate();
            default: return new DefaultPageTemplate();
        }
    }

    protected override void OnInitialized()
    {
        Dispatcher.Dispatch(new EditorFetchPageAction(Guid.Parse(Id)));
        base.OnInitialized();
    }

    public void HandleChanges(IComponentData componentData)
    {
        Dispatcher.Dispatch(new EditorHistoryPushAction(componentData));
    }

    public void Save()
    {
        Dispatcher.Dispatch(new EditorSavePageAction());
    }

    public void Share()
    {

NavigationManager.NavigateTo($"/pages/view/{EditorState.Value.Page.OwnerId}/
{EditorState.Value.Page.Key}");
    }

    public void ChangeName(string name)
    {
        Dispatcher.Dispatch(new EditorSetNameAction(name));
    }

    public void ChangeDisplayMode(EditorDisplayMode displayMode)
    {
        Dispatcher.Dispatch(new EditorSetDisplayModeAction(displayMode));
    }
}

<div class="h-full flex flex-col mx-2 my-2">
    @if (EditorState.Value.IsLoading)
    {
        <span>Loading...</span>
    }
    else
    {
        <Container Class="flex flex-row justify-between w-full mb-3">
            <div class="flex flex-row gap-2">
                <LabeledContainer Label="Page Name">
                    <MudInput T="string" @bind-
Value:get="EditorState.Value.Page.Name" @bind-Value:set="ChangeName" />
                </LabeledContainer>
                <HistoryController />
            </div>
            <div class="flex flex-row gap-2 items-center">
                <MudSelect T="EditorDisplayMode" @bind-
Value:get="EditorState.Value.DisplayMode"
                    @bind-Value:set="ChangeDisplayMode" Class="w-31">
                    @foreach (var item in Enum.GetValues<EditorDisplayMode>())
                    {
                        <MudSelectItem Value="item">@item</MudSelectItem>
```

```
                    }
                </MudSelect>
                <MudButton OnClick="Share" Color="Color.Primary"
Variant="Variant.Filled">Share</MudButton>
                <MudButton Disabled=EditorState.Value.Page.IsSaved
OnClick="Save" Color="Color.Primary"
                    Variant="Variant.Filled">Save</MudButton>
            </div>
        </Container>
        <div class="flex-auto overflow-y-scroll px-1 pb-1 fixed-sized-editor-
container">
            <EditorView>
                <Editor>
                    <EditorRenderer Component="context"
ComponentChanged="HandleChanges" />
                </Editor>
                <Preview>
                    <Providers QuizManager="QuizManager">
                        <ComponentRenderer Component="context" />
                    </Providers>
                </Preview>
            </EditorView>
        </div>
    }
</div>
```

Фрагмент коду перетворювача компонентного дерева в загальний вигляд

```
using StudentPortal.ComponentData.Abstractions;
using StudentPortal.ComponentData.Conversion.Abstractions;
using System.Collections;
using System.Diagnostics;
using System.Reflection;

namespace StudentPortal.ComponentData.Conversion;

public class DocumentConverter :
IComponentDataToDocumentConverter<IComponentData>
{
    public Document Convert(IComponentData component)
    {
        return Convert(component as object);
    }

    public Document Convert(object component)
    {
        var componentType = component.GetType();

        var objectProperties = componentType.GetProperties(BindingFlags.Instance
| BindingFlags.Public);

        var componentProperties =
objectProperties.Where(PropertyTypeIs(typeof(IComponentData)));
        var componentCollectionsProperties =
objectProperties.Where(PropertyTypeIsCollectionOf(typeof(IComponentData)));
        var other = objectProperties.Where(p => !componentProperties.Contains(p)
&& !componentCollectionsProperties.Contains(p));

        var components = new Dictionary<string, object>();
```

```
        return new Document()
        {
            Type = componentType.Name,
            Version = componentType.GetComponentVersion(),
            Components = GetPrimitiveComponents(componentProperties, component),
            ComponentCollections =
GetPrimitiveComponentLists(componentCollectionsProperties, component),
            Properties = GetPrimitiveProperties(other, component)
        };

        static Func<PropertyInfo, bool> PropertyTypeIs(Type type)
        {
            return (PropertyInfo p) =>
            {
                var propertyType = p.PropertyType;
                return propertyType.IsAssignableTo(type);
            };
        }

        static Func<PropertyInfo, bool> PropertyTypeIsCollectionOf(Type type)
        {
            return (PropertyInfo p) =>
            {
                var propertyType = p.PropertyType;

                if (!propertyType.TryGetCollectionType(out var elementType))
return false;

                return elementType.IsAssignableTo(type);
            };
        }
    }

    private Dictionary<string, Document>
GetPrimitiveComponents(IEnumerable<PropertyInfo> objectProperties, object
@object)
    {
        Dictionary<string, Document> components = [];

        foreach (var prop in objectProperties)
        {

Debug.Assert(prop.PropertyType.IsAssignableTo(typeof(IComponentData)));

            var name = prop.Name;

            var componentValue = prop.GetValue(@object) as IComponentData;

            if (componentValue is null) continue;

            var value = Convert(componentValue);

            components.Add(name, value);
        }

        return components;
    }

    private Dictionary<string, Document[]>
GetPrimitiveComponentLists(IEnumerable<PropertyInfo> objectProperties, object
@object)
    {
        var componentCollections = new Dictionary<string, Document[]>();
```

```
        foreach (var prop in objectProperties)
        {
            {
                Debug.Assert(prop.PropertyType.TryGetCollectionType(out var
elementType));

Debug.Assert(elementType.IsAssignableTo(typeof(IComponentData)));
            }

            var name = prop.Name;

            var collection = prop.GetValue(@object) as IEnumerable;
            var componentCollection = collection?.Cast<IComponentData>();

            if (componentCollection is null) continue;

            var value = componentCollection.Select(Convert).ToArray();

            componentCollections.Add(name, value);
        }

        return componentCollections;
    }

    private Dictionary<string, object>
GetPrimitiveProperties(IEnumerable<PropertyInfo> objectProperties, object
@object)
    {
        var props = new Dictionary<string, object>();

        foreach (var prop in objectProperties)
        {
            var name = prop.Name;
            var value = prop.GetValue(@object);

            var convertedValue = TypeConverter.ConvertToJsonPrimitive(value);

            if (convertedValue is null) continue;

            props.Add(name, convertedValue);
        }

        return props;
    }
}
```

Фрагмент коду обрахунку балів

```
namespace StudentPortal.ComponentData.Quizzes;

public class DefaultScoreCalculator : IQuizScoreCalculator,
IAnswerVisitor<double?>
{
    public static readonly DefaultScoreCalculator Instance = new
DefaultScoreCalculator();
    public double? GetScore(IAnswer<IQuestionDeclaration> answer)
    {
        return answer.Accept(this);
    }
```

```
    public double? Visit(VarianceQuestionAnswer answer)
    {
        var (question, answers) = answer;

        var correctAnswersCount = answers.Union(question.Answers).Count();
        var totalAnswersCount = answers.Count;
        var wrongAnswersCount = Math.Max(correctAnswersCount -
totalAnswersCount, 0);

        return question.Answers.Count switch
        {
            0 => question.MaxScore,
            1 => correctAnswersCount != 0 ? question.MaxScore : 0,
            _ => (correctAnswersCount - wrongAnswersCount) /
(double)question.Answers.Count * question.MaxScore
        };
    }

    public double? Visit(OpenAnswerQuestionAnswer answer)
    {
        return null;
    }
}


public static class AnswerScoreExtensions
{
    public static double? GetScore(this IAnswer<IQuestionDeclaration> answer)
    {
        return answer.Accept(DefaultScoreCalculator.Instance);
    }
}
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ВЕБЗАСТОСУНОК ДЛЯ СТВОРЕННЯ ТА ВИКОРИСТАННЯ ЦИФРОВИХ НАВЧАЛЬНИХ МАТЕРІАЛІВ

Виконав: Мазний С.В.

Керівник: Доцент кафедри ПЗКС, к.т.н., доцент, Заболотня Тетяна Миколаївна

Київ – 2025

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: забезпечити гнучку роботу з цифровими навчальними матеріалами шляхом створення відповідного ПЗ

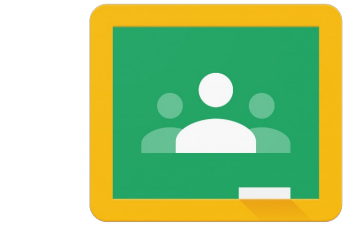
Завдання:

- Проаналізувати ринок ПЗ цифрових навчальних матеріалів
- Порівняти існуючі рішення, виявити їхні переваги та недоліки
- Визначити функціональні та нефункціональні вимоги до ПЗ
- Підготувати дизайн користувацького інтерфейсу застосунку
- Обґрунтувати вибір стеку технологій для розроблення застосунку
- Реалізувати вебзастосунок для створення, редагування та використання навчальних матеріалів
- Протестувати систему на відповідність встановленим вимогам
- Навести шляхи подальшого вдосконалення ПЗ

АКТУАЛЬНІСТЬ

- Зростає попит на цифрові інструменти в освіті
- Існуючі рішення часто незручні або обмежені
- Потреба у вебзастосуноку для роботи з цифровими навчальними матеріалами
- Інтеграція сучасних технологій у навчальний процес
- Недостатня гнучкість наявних платформ

НАЯВНІ АНАЛОГИ



Застосунок
«Google Classroom»

Застосунок
«Moodle LMS»

Застосунок
«Canva for Education»

ФУНКЦІОНАЛЬНІ ВИМОГИ ДО РОЗРОБЛЮВАНОГО ПЗ

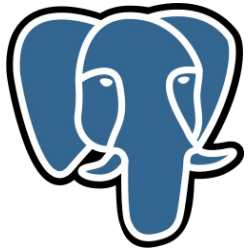
Вебзастосунок повинен забезпечувати такі основні вимоги:

1. Створення навчальних сторінок з текстом, зображеннями, відео та тестами
2. Редагування контенту через візуальний редактор із збереженням змін
3. Автоматична перевірка тестів з однією чи кількома правильними відповідями
4. Доступ студентів до матеріалів через зручний, інтуїтивний інтерфейс
5. Можливість вибору шаблону при створенні нового матеріалу (текст, тест тощо)
6. Реєстрація користувачів, авторизація та підтримка ролей (викладач / студент)

ОБРАНІ ЗАСОБИ РЕАЛІЗАЦІЇ

ASP.NET Core

Angular



PostgreSQL

MongoDB

TailwindCSS

Blazor

АРХІТЕКТУРА ВЕБЗАСТОСУНКУ

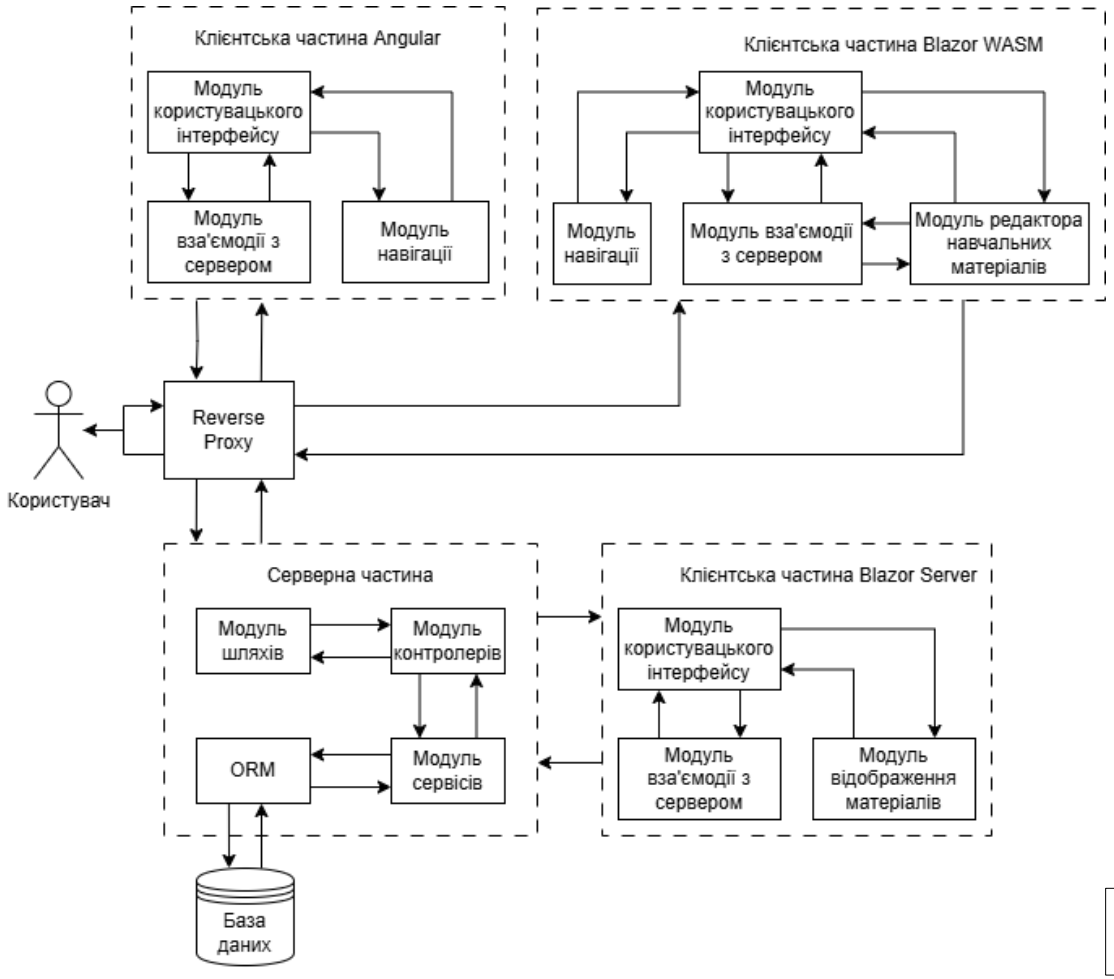
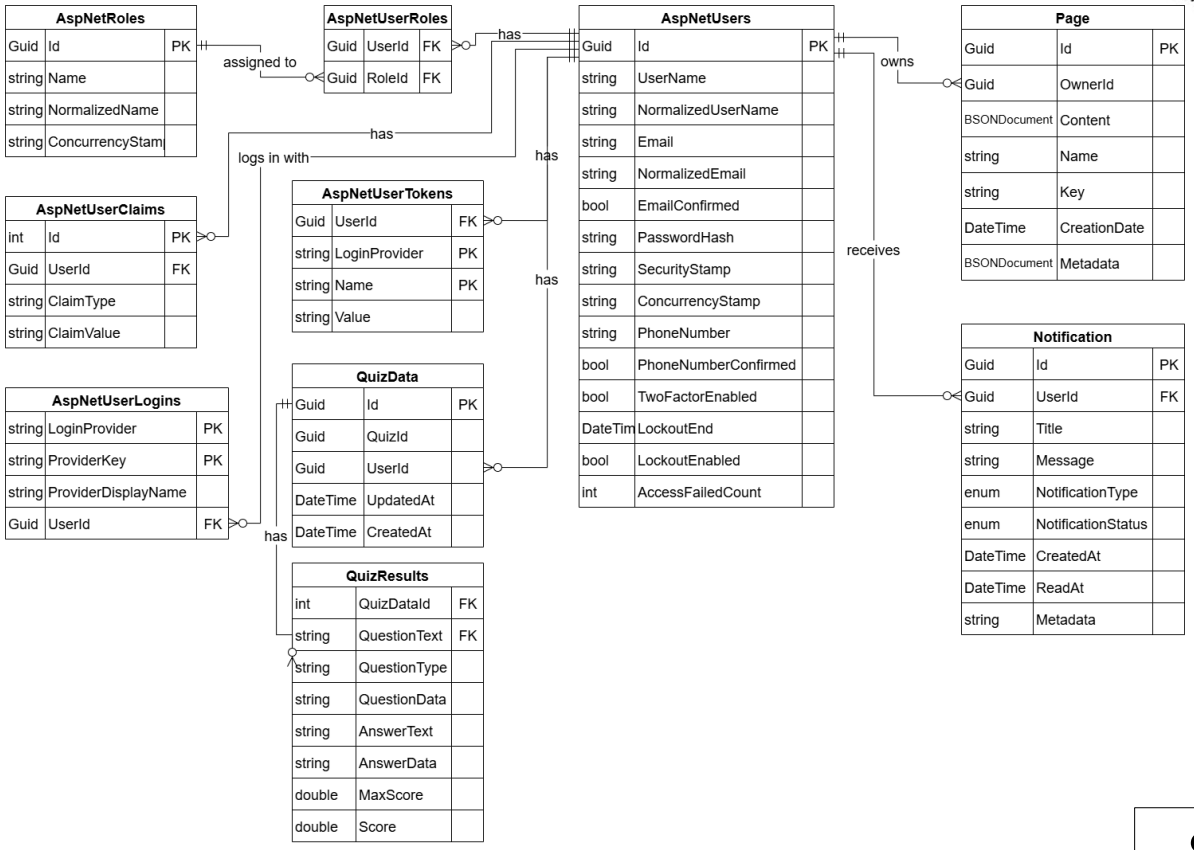
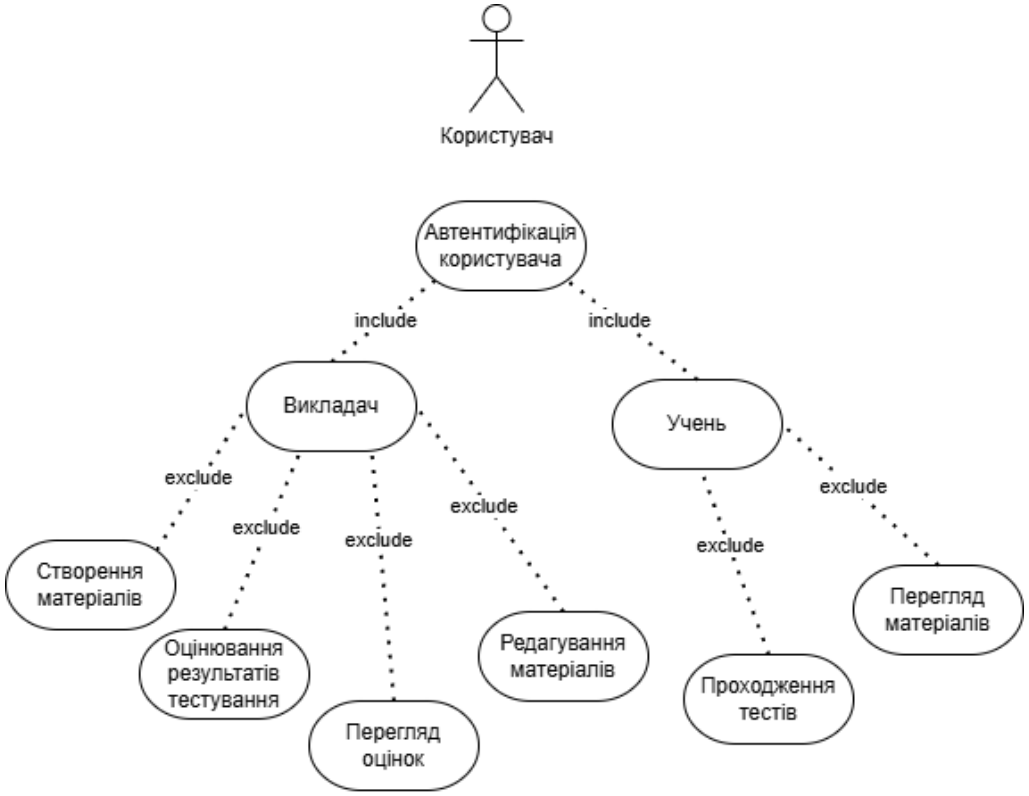


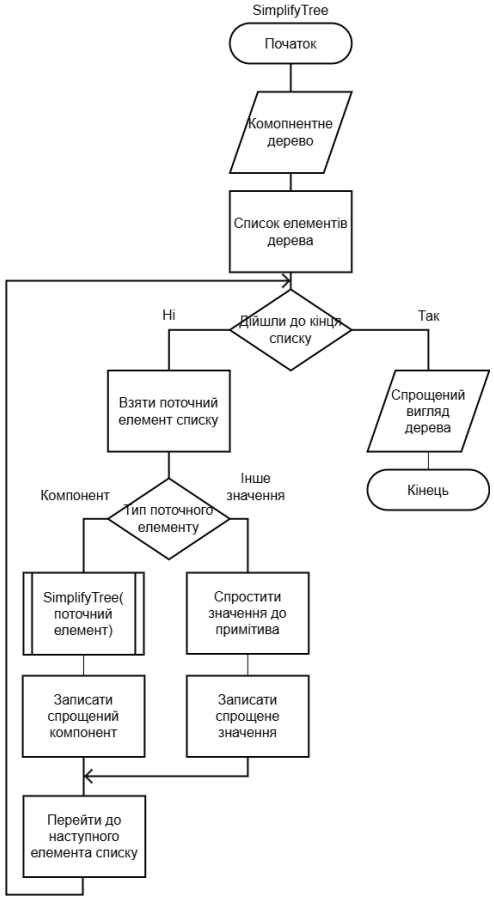
СХЕМА БАЗИ ДАНИХ ЗАСТОСУНКУ



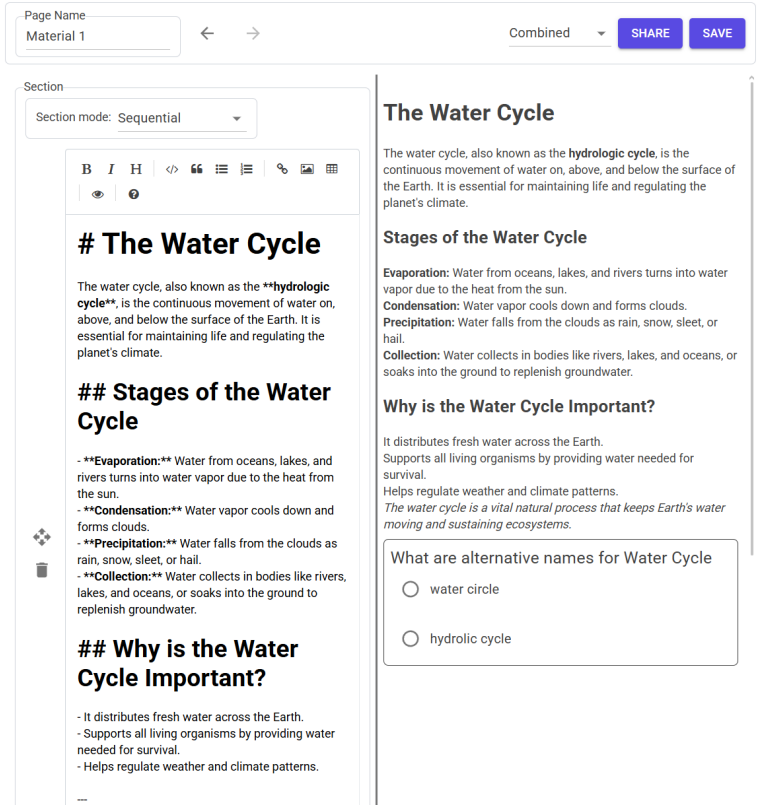
ДІАГРАМА ПРЕЦЕДЕНТІВ



БЛОК-СХЕМА ПЕРЕТВОРЕННЯ КОМПОНЕНТНОГО ДЕРЕВА



СТОРІНКА РЕДАКТОРА



ТЕСТУВАННЯ ЗАСТОСУНКУ

Застосунок було протестовано на трьох рівнях: бізнес-логіка, API та інтерфейс користувача.

Рівень тестування	Основні напрямки перевірки	Інструменти / методи
Unit-тестування	• Реєстрація та авторизація • Управління навчальними матеріалами • Автоматична перевірка тестів • Система ролей	Вбудовані фреймворки для .NET
API-тестування	• Формат і статус-коди відповідей • Обробка помилок • Навантаження • Безпека та автентифікація	Postman, логування запитів
Тестування інтерфейсу	• Візуальна відповідність макетам • Навігація та валідація форм • Інтерактивність та повідомлення • Кросбраузерність	Ручне тестування, браузери Chrome, Firefox, Edge

НАПРЯМКИ ПОДАЛЬШОГО РОЗВИТКУ ПРОДУКТУ

- Інтеграція з LMS (Moodle, Google Classroom) для синхронізації курсів і прогресу студентів.
- Додавання нових типів завдань: відкриті запитання, завантаження файлів, інтерактивні вправи.
- Підключення аналітики для оцінки успішності та формування персональних рекомендацій.
- Підтримка офлайн-доступу з подальшою синхронізацією при підключенні до мережі.
- Розширення мовної підтримки для доступності різним категоріям користувачів.

ВИСНОВКИ

- Проаналізовано існуючи аналоги на ринку.
- Визначено функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення.
- Обрано програмні засоби для реалізації та обгрунтовано їх вибір.
- Розроблено архітектуру та дизайн вебзастосунку.
- Реалізовано поставлені до вебзастосунку вимоги.
- Проведене тестування розробленого вебзастосунку, виявлені помилки виправлено
- Визначено шляхи подальшого покращення продукту

УНІКАЛЬНІСТЬ

Автор
Мазний Степан Валерійович

Науковий керівник / Експерт
Заболотня Тетяна Миколаївна

підрозділ
ФПМ, К-ра програмного забезпечення комп'ютерних систем

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1

15/16

ДЯКУЮ ЗА УВАГУ

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ СТВОРЕННЯ ТА ВИКОРИСТАННЯ ЦИФРОВИХ НАВЧАЛЬНИХ МАТЕРІАЛІВ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Степан МАЗНИЙ

2024

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Вебзастосунок «StudentPortal» для створення та використання цифрових навчальних матеріалів, дозволяє викладачам створювати, редагувати та перевіряти навчальний контент, включно з тестовими завданнями, а студентам – зручно його опрацьовувати. Реалізовано з використанням Angular та Blazor для клієнтської частини, ASP.NET Core для серверної логіки. Для зберігання даних застосовано PostgreSQL і MongoDB.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено правильність, надійність та стабільність роботи вебзастосунку StudentPortal відповідно до функціональних та нефункціональних вимог. Тестування дозволяє виявити можливі помилки, оцінити зручність інтерфейсу та підтвердити готовність продукту до використання в освітньому середовищі.

У межах тестування перевіряється:

- коректність реєстрації, авторизації та розмежування прав доступу;
- створення, редагування та збереження навчальних матеріалів;
- стабільність роботи візуального редактора;
- функціонування тестів та автоматичної перевірки відповідей;
- відображення матеріалів у ролі студента;
- робота з шаблонами сторінок;
- взаємодія клієнтської частини з API;
- відповідність бази даних вимогам до структури та збереження інформації;
- загальна продуктивність, надійність і стабільність системи.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування вебзастосунку StudentPortal проводилось на кількох рівнях для перевірки надійності, стабільності та відповідності функціональним вимогам. Основними методами були модульне тестування, перевірка API та тестування інтерфейсу.

Unit-тести охоплювали ключову бізнес-логіку – реєстрацію, авторизацію, керування навчальними матеріалами, перевірку тестових відповідей та розмежування доступу за ролями.

API тестувалося окремо з метою перевірки правильності відповідей, обробки помилок, автентифікації та захисту даних. Тестування інтерфейсу проводилось вручну та частково автоматизовано, із фокусом на коректну навігацію, валідацію форм, реакцію на дії користувача та адаптивність дизайну.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Модульне тестування здійснювалось за допомогою xUnit у середовищі .NET. Для кожного модуля створювались окремі тести, що перевіряли логіку обробки даних та граничні випадки.

API перевірялось вручну через Postman. Тестувались як стандартні сценарії, так і некоректні запити – на відсутність параметрів, неправильні типи даних або спроби несанкціонованого доступу.

Інтерфейс тестувався у браузерах Chrome, Firefox і Edge для перевірки кросбраузерної сумісності. Тестувались інтерактивні елементи, навігація, повідомлення про помилки та адаптивність верстки.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ СТВОРЕННЯ ТА ВИКОРИСТАННЯ ЦИФРОВИХ НАВЧАЛЬНИХ МАТЕРІАЛІВ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Степан МАЗНИЙ

2025

ЗМІСТ

1. Опис структури вебзастосунку

Вебзастосунок «StudentPortal» призначений для організації цифрового навчального процесу, зокрема створення, редагування, виконання та перевірки навчальних матеріалів і тестових завдань. Система динамічно змінює доступну функціональність залежно від ролі користувача (викладач або студент), забезпечуючи гнучкість у використанні.

До складу застосунку входять основні сторінки та модулі:

- головна сторінка;
- сторінка реєстрації;
- сторінка авторизації;
- сторінка списку матеріалів викладача;
- сторінка редактору;
- сторінка проходження тесту або перегляду матеріалу;
- сторінка результатів виконаного завдання (для студента);
- сторінка перегляду результатів студентів (для викладача).

2. Головна сторінка

Для початку роботи з вебзастосунком StudentPortal користувач потрапляє на головну сторінку, яка доступна без попередньої авторизації (рис. 1). На цій сторінці представлені два основні варіанти дій:

Sign In – перехід до форми входу в обліковий запис;

Join Now – перехід до форми реєстрації нового користувача.

Ці дії є обов’язковими для отримання доступу до функціоналу системи.

Натискаючи Sign In, користувач переходить на сторінку авторизації, де необхідно ввести зареєстровану електронну адресу та пароль для входу в систему.

Натискаючи Join Now, відкривається сторінка реєстрації нового користувача, де потрібно обрати роль (студент або викладач) та заповнити реєстраційну форму. Тільки після успішної авторизації або реєстрації користувач отримує доступ до персоналізованої головної сторінки з відповідним функціоналом згідно з його роллю в системі.

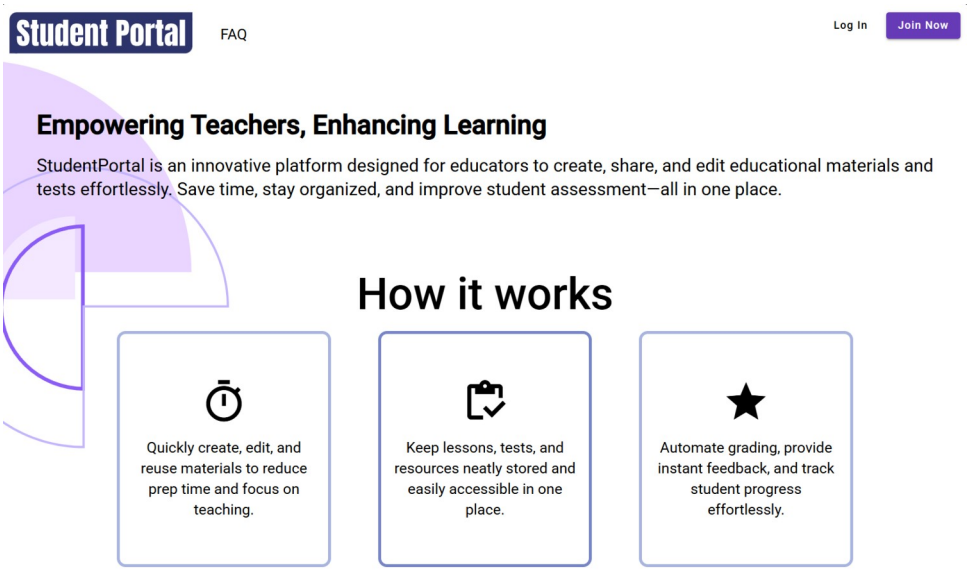


Рис. 1. Головна сторінка вебзастосунку

3. Процедура авторизації користувача

Для створення облікового запису в системі «StudentPortal» користувач повинен перейти на сторінку реєстрації (рис. 2), яка відкривається після натискання кнопки Join Now на головній сторінці.

На сторінці реєстрації представлено такі обов’язкові поля для заповнення:

- First Name та Last Name – ім’я та прізвище користувача.
- Email – чинна електронна адреса, яка буде використовуватися для входу в систему.

- Password – пароль, що має містити щонайменше 8 символів, а також базові правила для збільшення його складності.
- Confirm Password – повторне введення пароля для перевірки правильності.
- Роль – вибір ролі користувача: Student (студент) або Teacher (викладач).

Після заповнення всіх полів необхідно натиснути кнопку «Submit» для завершення процесу реєстрації. У випадку помилок (наприклад, невалідна електронна адреса або невідповідність паролів), система виведе відповідні повідомлення про помилку, які потрібно виправити.

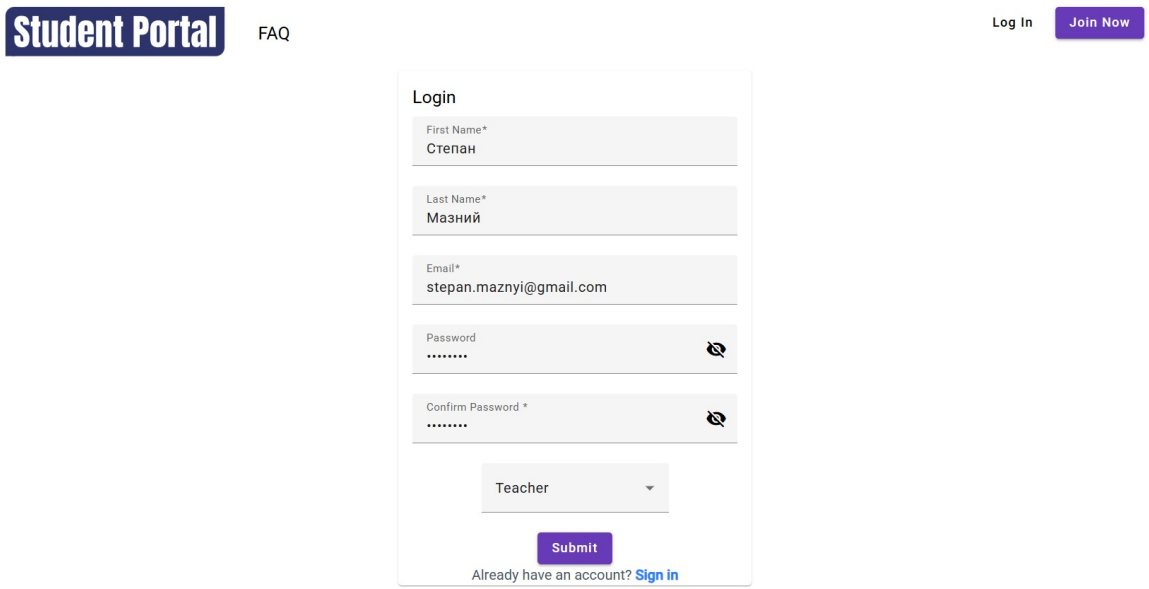


Рис. 2. Сторінка реєстрації користувача

Якщо користувач вже має створений обліковий запис, він може перейти на сторінку авторизації натиснувши на посилання «Sing in» або на кнопку «Log In» згори сторінки.

Сторінку авторизації надано на рис. 3.

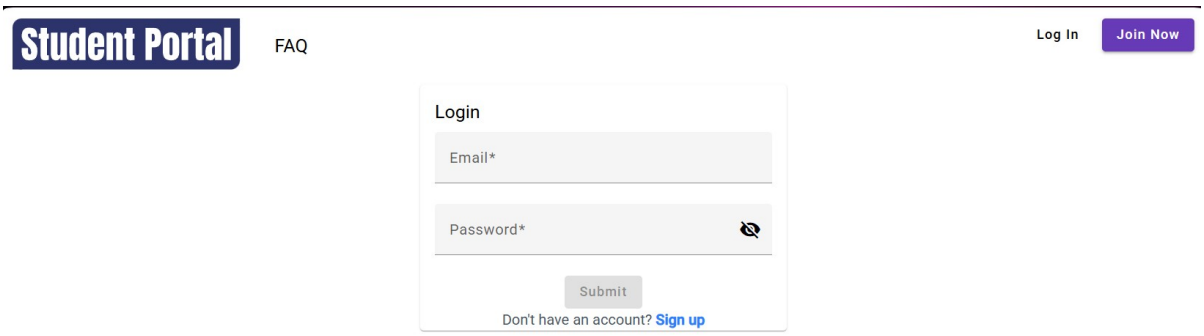


Рис. 3. Сторінка авторизації

4. Сторінка списку матеріалів викладача

Після успішної реєстрації або входу в систему користувач автоматично перенаправляється на головну сторінку з відображенням його акаунту(рис. 4) та вийти з нього в меню керування.

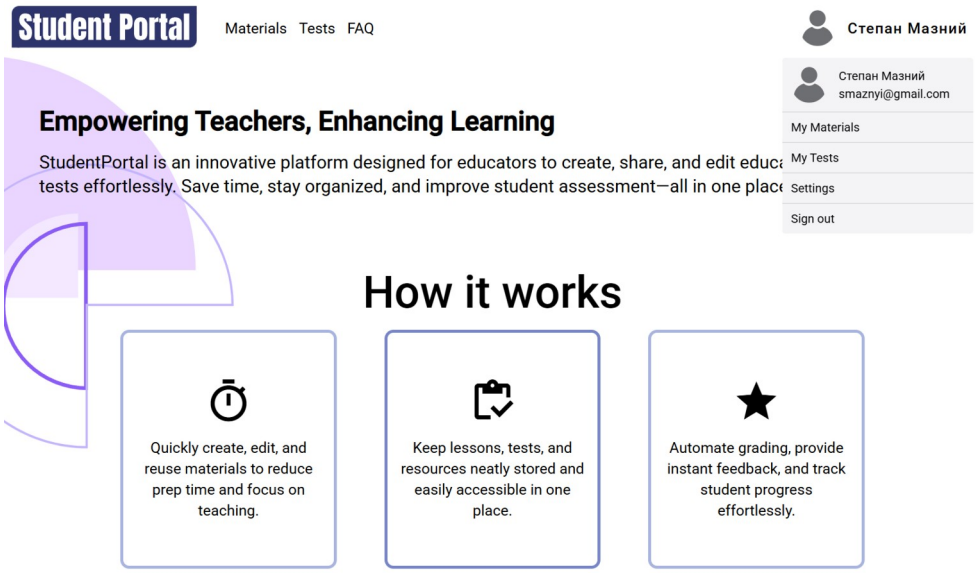


Рис. 4. Сторінка авторизації

Якщо було обрано роль «Викладач», користувач отримує доступ до інтерфейсу керування навчальними матеріалами (рис. 5).

Сторінка керування матеріалами надає зручний візуальний інтерфейс у вигляді карток. Кожна картка представляє окремий навчальний матеріал

із назвою та іконкою. При натисканні на картку відкривається відповідний матеріал для перегляду або редагування.

У правому верхньому куті кожної картки розташовано меню дій (іконка з трьома крапками), яке дозволяє:

- відкрити матеріал;
- перейменувати назву;
- видалити матеріал;
- скопіювати існуючий матеріал.

Для створення нового навчального матеріалу передбачено кнопку з іконкою "+", розташовану у верхньому правому куті сторінки. При її натисканні відкривається вікно вибору шаблону (рис. 6), після вибору якого новий матеріал з’являється у загальному списку.

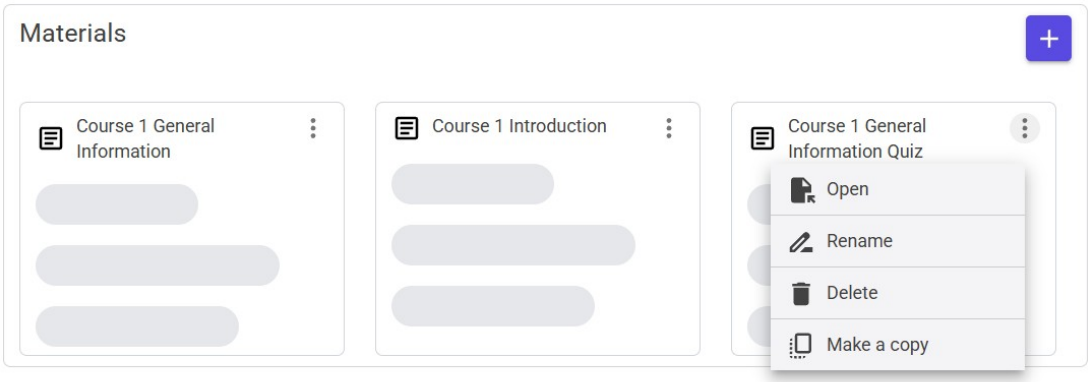


Рис. 5. Сторінка керування матеріалами викладача

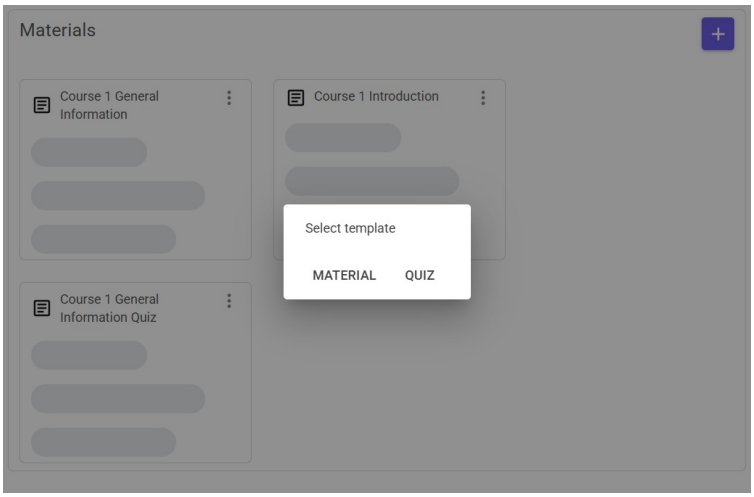


Рис. 6. Сторінка керування матеріалами викладача

5. Модуль редактора

Сторінка редагування (рис. 7) призначена для створення та зміни навчального контенту. Вона містить текстовий редактор із панеллю форматування, підтримує додавання мультимедіа та перегляд результату в реальному часі.

Назва матеріалу задається у верхній лівій частині у полі Page Name. Поруч – кнопки скасування та повторення дій.

Режим відображення змінюється через випадаючий список:

- Editor – лише редактор.
- Preview – попередній перегляд.
- Combined – редактор і перегляд одночасно.

Праворуч розташовані кнопки

- Share – відкриває посилання для перегляду матеріалу.
- Save – зберігає зміни.

Основну частину займає область редагування, де можна форматувати текст, вставляти зображення, відео та створювати інтерактивний контент.

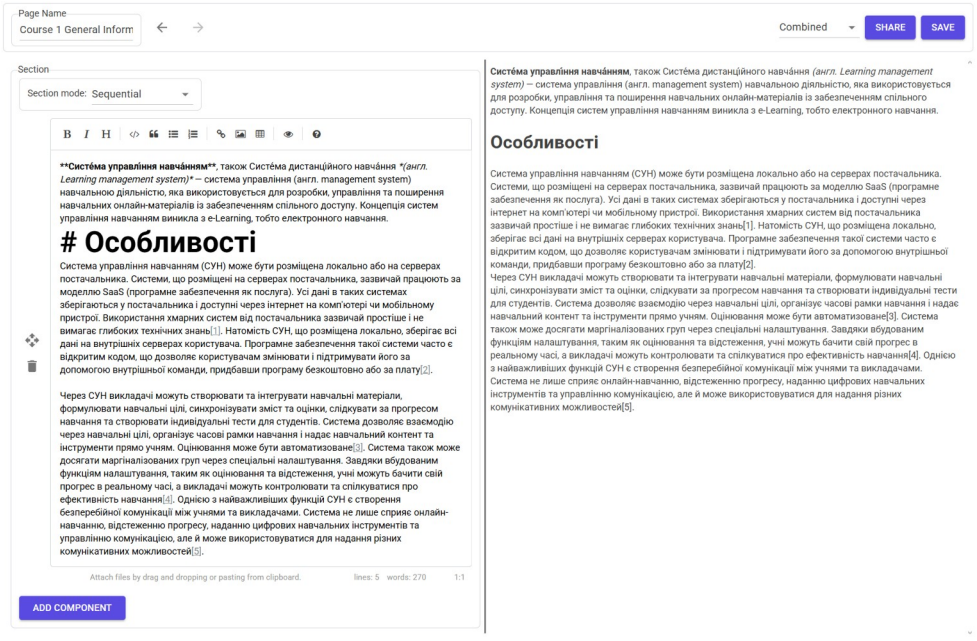


Рис. 7. Сторінка редактора

6. Модуль відображення матеріалів

Сторінка перегляду матеріалів (рис. 8-9) дозволяє перегляд учнями створених матеріалів та проходження тестів.

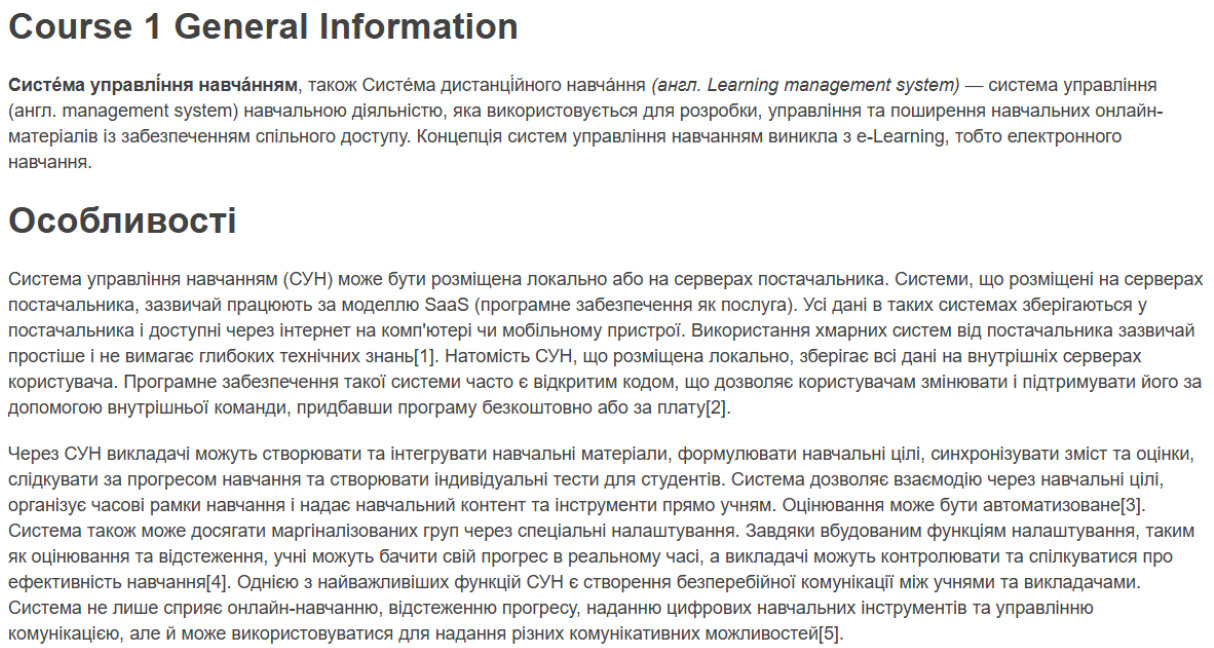


Рис. 8. Сторінка перегляду матеріалів, матеріал

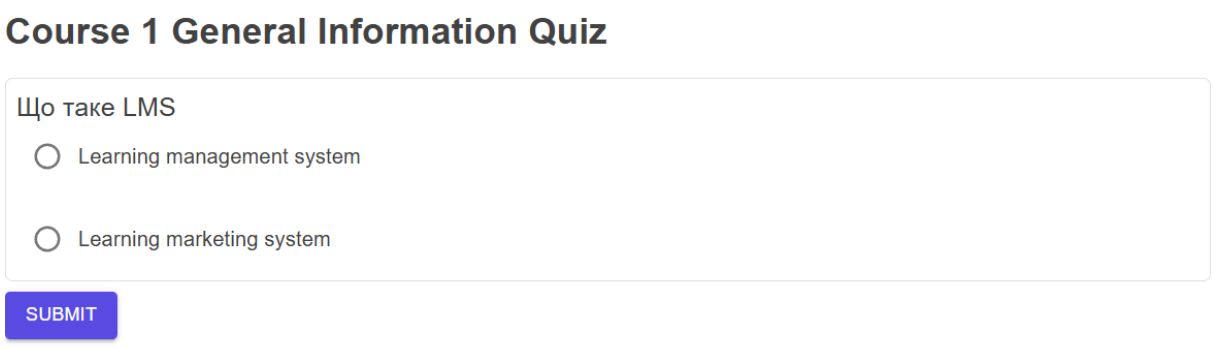


Рис. 9. Сторінка перегляду матеріалів, тест

7. Модуль відображення матеріалів

Сторінка перегляду результатів тестування студентів (рис. 10) призначена для виставлення оцінок за тести та перегляд даних тестування.

Вона містить таблицю зі студентами, які пройшли тест з оцінкою та кнопкою «Details», яка дозволяє більш детально оцінити роботу.

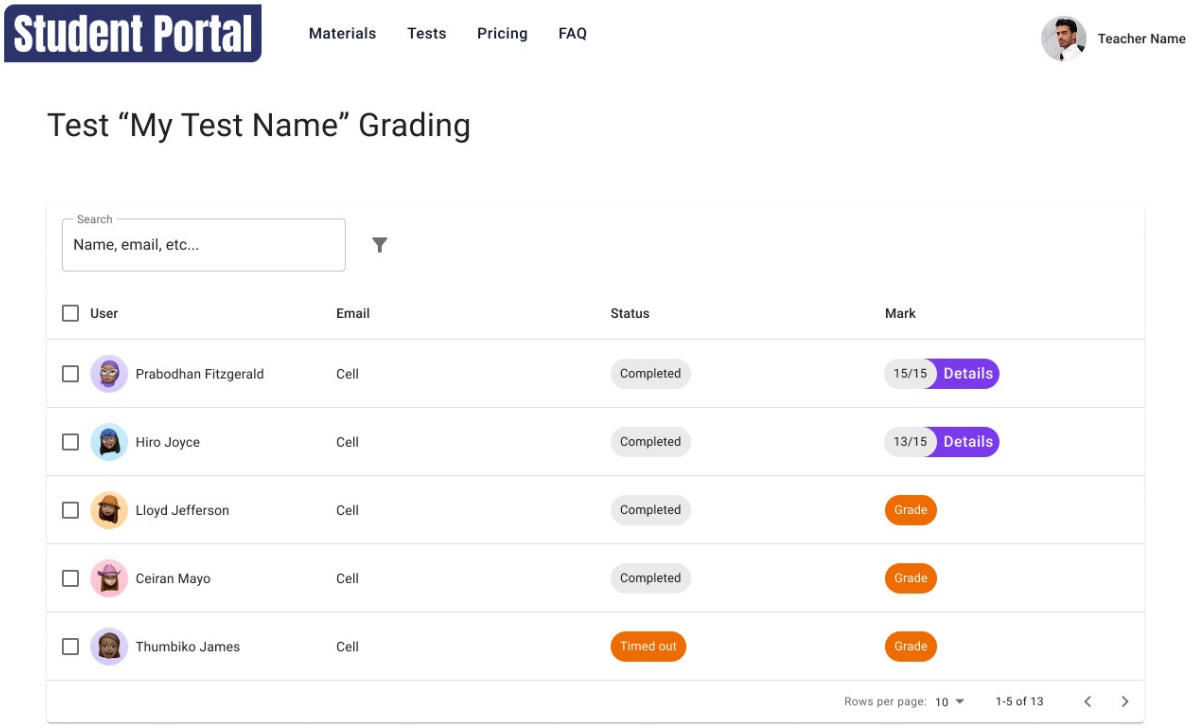


Рис. 10. Сторінка перегляду результатів студентів (для викладача)