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

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

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

спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок для організації роботи автошколи

Виконав студент IV курсу, групи ІП-13
(шифр групи)
Карамян Вартан Суренович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник асистент, Зарічковий О.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант ст. викл., Вітковська І. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент асистент Альбрехт Й. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Студент _____
(підпис)

Київ – 2025

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

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Карамяну Вартану Суреновичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для організації роботи автошколи
керівник проєкту Зарічковий Олександр Анатолійович, асистент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с

2. Термін подання студентом проєкту «16» червня 2025 року

3. Вихідні дані до проєкту: технічне завдання

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

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____

2) Схема бази даних _____

3) Креслення вигляду екранних форм _____

_____

_____

_____

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «15» березня 2025 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв’язання задачі	29.03.2025	
3	Постановка та формалізація задачі	12.04.2025	
4	Розробка інформаційного забезпечення	19.04.2025	
5	Алгоритмізація задачі	23.04.2025	
6	Обґрунтування вибору використаних технічних засобів	26.04.2025	
7	Розробка програмного забезпечення	20.05.2025	
8	Налагодження програми	25.05.2025	
9	Виконання графічних документів	27.05.2025	
10	Оформлення пояснювальної записки	01.06.2025	
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	14.06.2025	

Студент _____ Вартан КАРАМЯН

(підпис) (ініціали, прізвище)

Керівник _____ Олександр ЗАРІЧКОВИЙ

(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 69 таблиць, 38 рисунків, 9 формул та 23 джерел – загалом 99 сторінок.

Дипломний проєкт присвячений розробці вебзастосунку для організації роботи автошколи.

Метою дипломного проєкту є підвищення ефективності роботи автошколи шляхом консолідації декількох розрізнених застосунків у єдину інтегровану платформу.

У першому розділі проведено аналіз предметної області, проаналізовано існуючі програмні продукти та технічні рішення, описано основні бізнес-процеси.

Другий розділ присвячений визначенню варіантів використання, розробці функціональних та нефункціональних вимог, аналізу системних вимог та економічних показників, постановці задачі.

У третьому розділі розроблено архітектуру програмного забезпечення, обґрунтовано вибір засобів розробки, здійснено конструювання та розроблення програмного забезпечення, проаналізовано безпеку даних.

Четвертий розділ складається з аналізу якості програмного забезпечення, опису процесів тестування та опису контрольного прикладу для усіх ролей.

У п’ятому розділі описано особливості розгортання та супроводу програмного забезпечення.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, МІКРОСЕРВІС, АВТОМАТИЗАЦІЯ, КОНСОЛІДАЦІЯ, АВТОШКОЛА, СТУДЕНТ, ВИКЛАДАЧ, АДМІНІСТРАТОР, PYTHON, TYPESCRIPT, БАЗА ДАНИХ.

ABSTRACT

The explanatory note of the diploma project consists of five sections and contains 69 tables, 38 figures, 9 formulas and 23 sources – in total 99 pages.

The purpose of the diploma project is to develop a web application for organizing the operations of a driving school.

The aim is to enhance the school's efficiency by consolidating several separate applications into a single, integrated platform.

The first chapter presents an analysis of the subject area, a review of existing software products and technical solutions, and a description of the main business processes.

The second chapter is dedicated to defining use cases, developing functional and non-functional requirements, analyzing system requirements and economic indicators, and formulating the project task.

The third chapter outlines the software architecture, justifies the choice of development tools, details the construction and development of the application, and analyzes data security.

The fourth chapter includes an analysis of software quality, a description of testing processes, and a sample use case for all user roles.

The fifth chapter describes the deployment and maintenance of the software.

KEYWORDS: WEB APPLICATION, MICROSERVIS, AUTOMATION, CONSOLIDATION, DRIVING SCHOOL, STUDENT, TEACHER, ADMINISTRATOR, PYTHON, TYPESCRIPT, DATABASE.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-1313.045440.01.91	Технічне завдання	27	
3	А4	КПІ.ІП-1313.045440.02.81	Пояснювальна записка	101	
4	А4	КПІ.ІП-1313.045440.03.12	Текст програми	25	
5	А4	КПІ.ІП-1313.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-1313.045440.05.34	Керівництво користувача	29	
7	А3	КПІ.ІП-1313.045440.06.99	Графічний матеріал	4	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-1313.045440.00.90		
Розроб.		Карамян В.С.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Зарічковий О.А.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-13	
Н.контр.		Вітковська І.І.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ АВТОШКОЛИ

Технічне завдання

КПІ.ІП-1313.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олександр ЗАРІЧКОВИЙ

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Вартан КАРАМЯН

Київ – 2025

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебзастосунок для організації роботи автошколи.

Галузь застосування:

Наведене технічне завдання поширюється на розробку вебзастосунку для організації роботи автошколи «DreamDrive» [045440] , котра використовується для цифровізації основних бізнес-процесів автошколи, зокрема управління навчанням, тестуванням, розкладом занять, оплатами та призначена для адміністраторів, викладачів і студентів автошкіл.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

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

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для цифровізації та автоматизації ключових бізнес-процесів автошколи.

Метою розробки є підвищення ефективності роботи автошколи шляхом консолідації кількох розрізнених цифрових рішень в єдину інтегровану платформу.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– головна сторінка автошколи (рисунок 4.1);

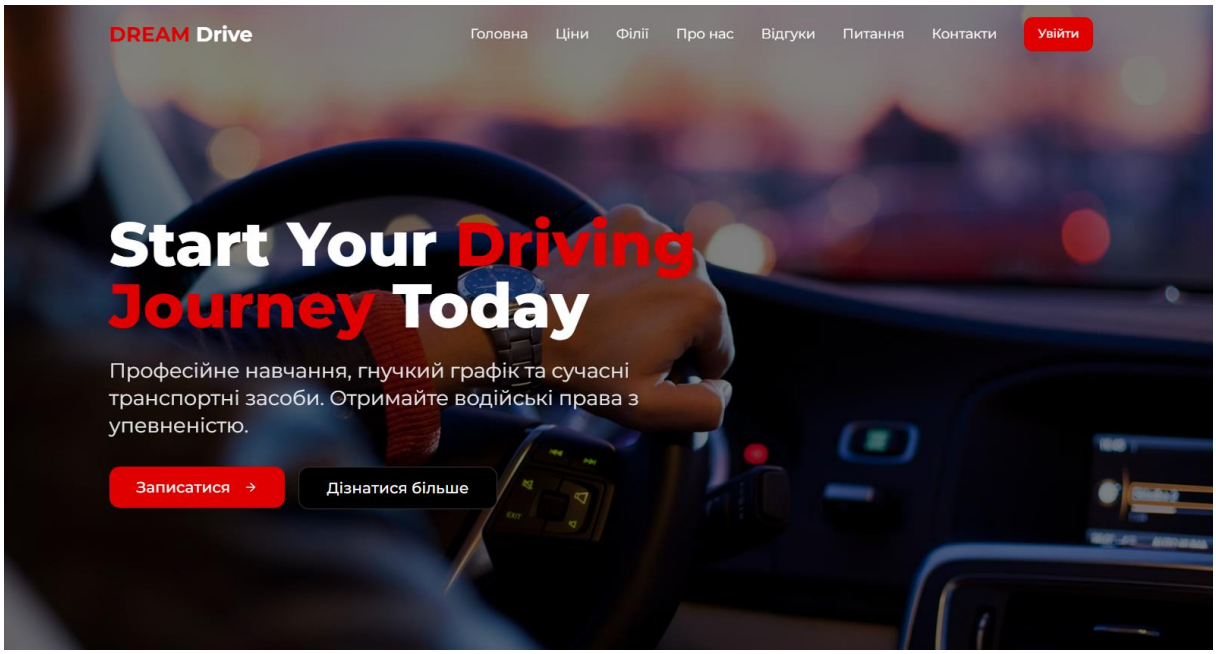


Рисунок 4.1 – Прототип головної сторінки

– сторінка для входу в систему (рисунок 4.2);

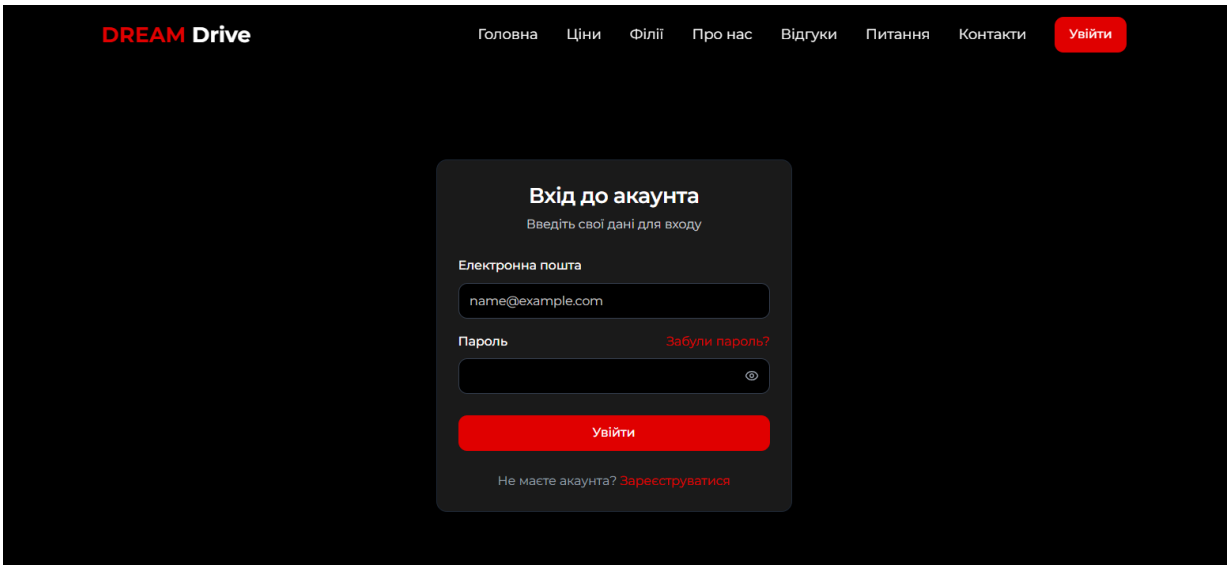


Рисунок 4.2 – Прототип сторінки для входу в систему

– сторінка для реєстрації в системі (рисунок 4.3);

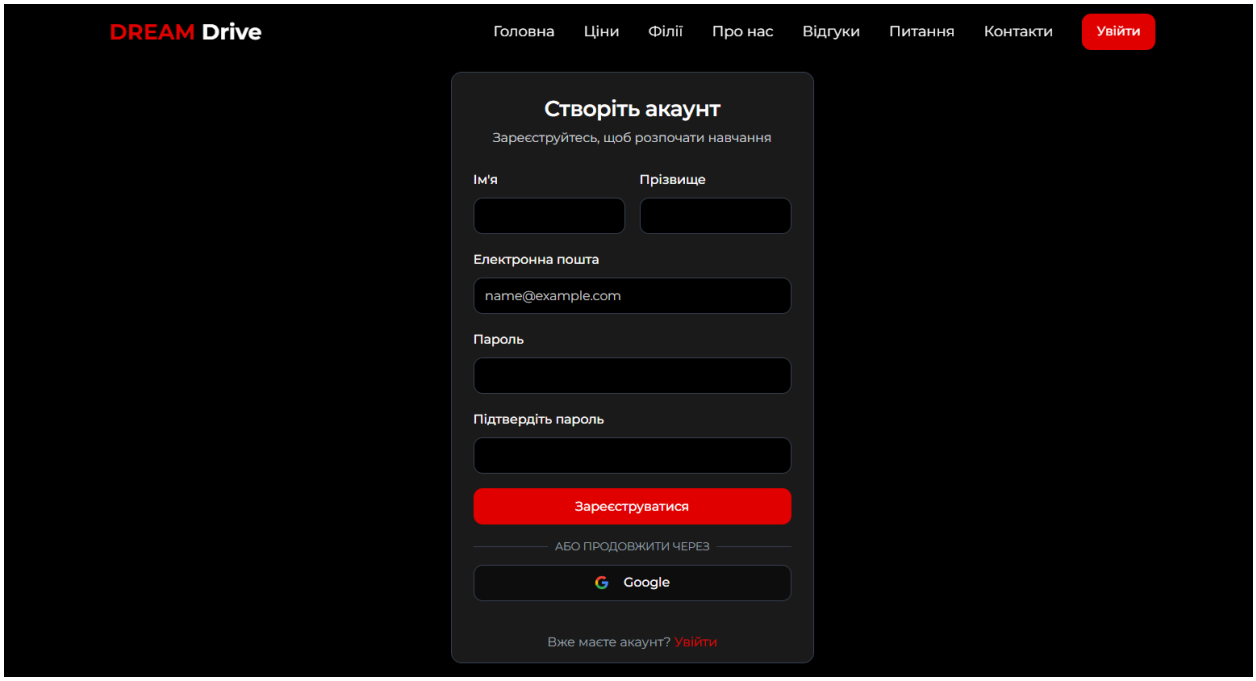


Рисунок 4.3 – Прототип сторінки для реєстрації в системі

– сторінка для перегляду та редагування профілю (рисунок 4.4);

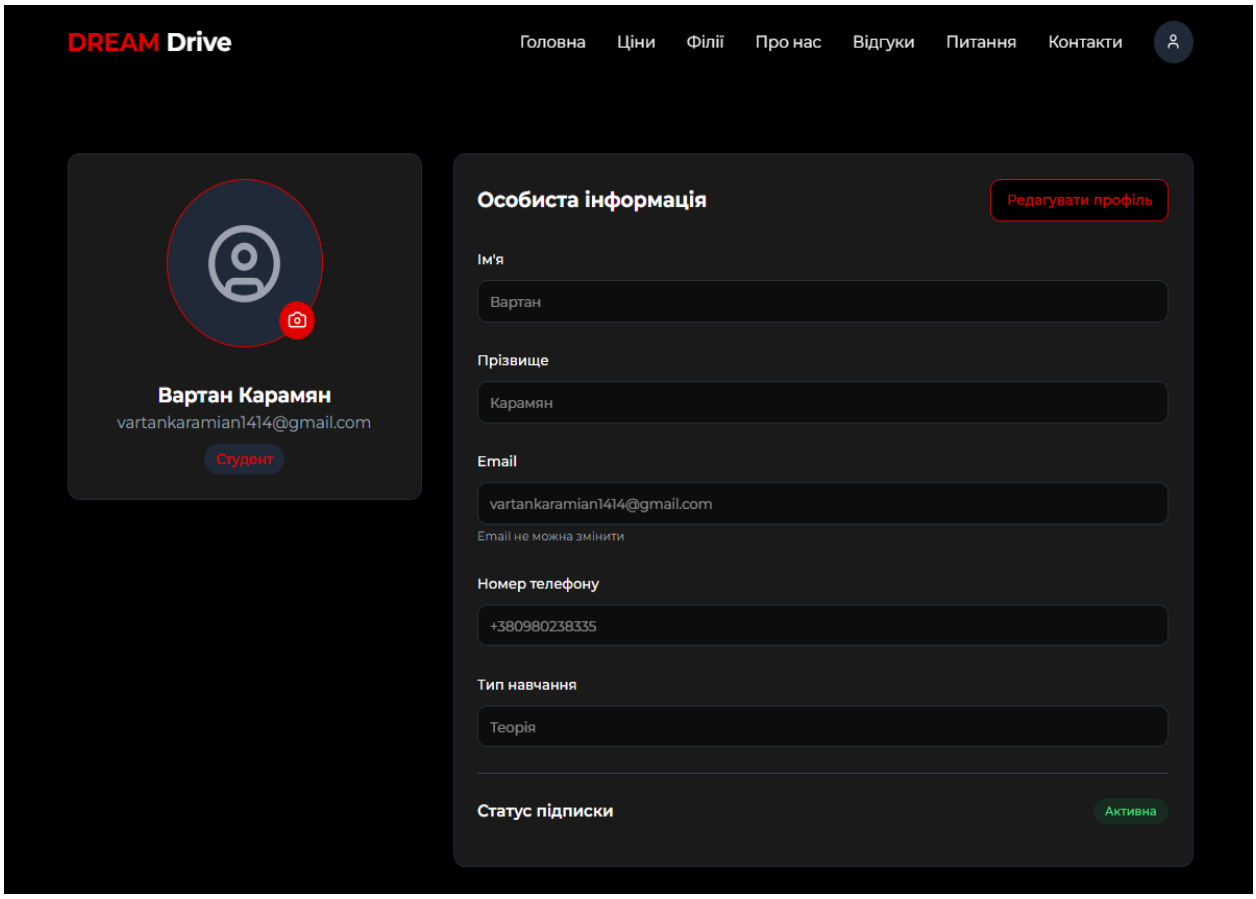


Рисунок 4.4 – Прототип сторінки для перегляду та редагування профілю

– панель керування студента (рисунок 4.5);

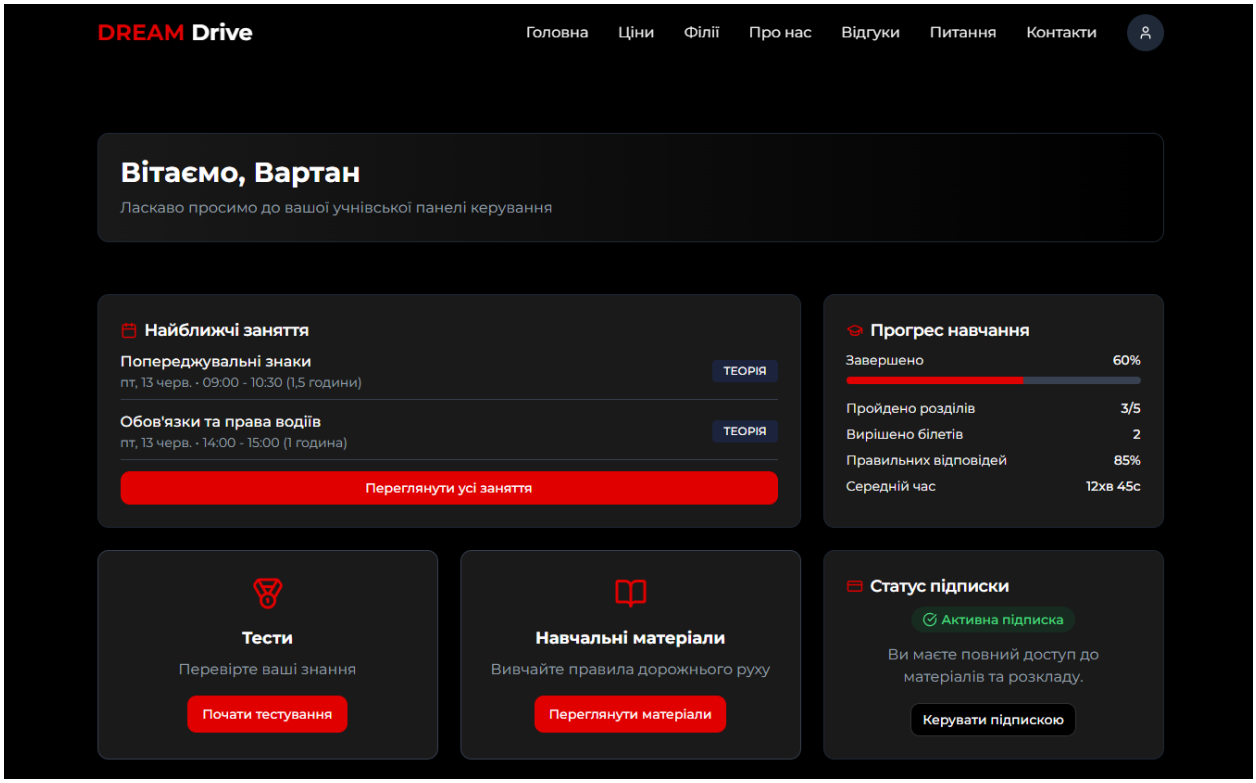


Рисунок 4.5 – Прототип сторінки панелі керування студента

– сторінка платежів студента (рисунок 4.6);

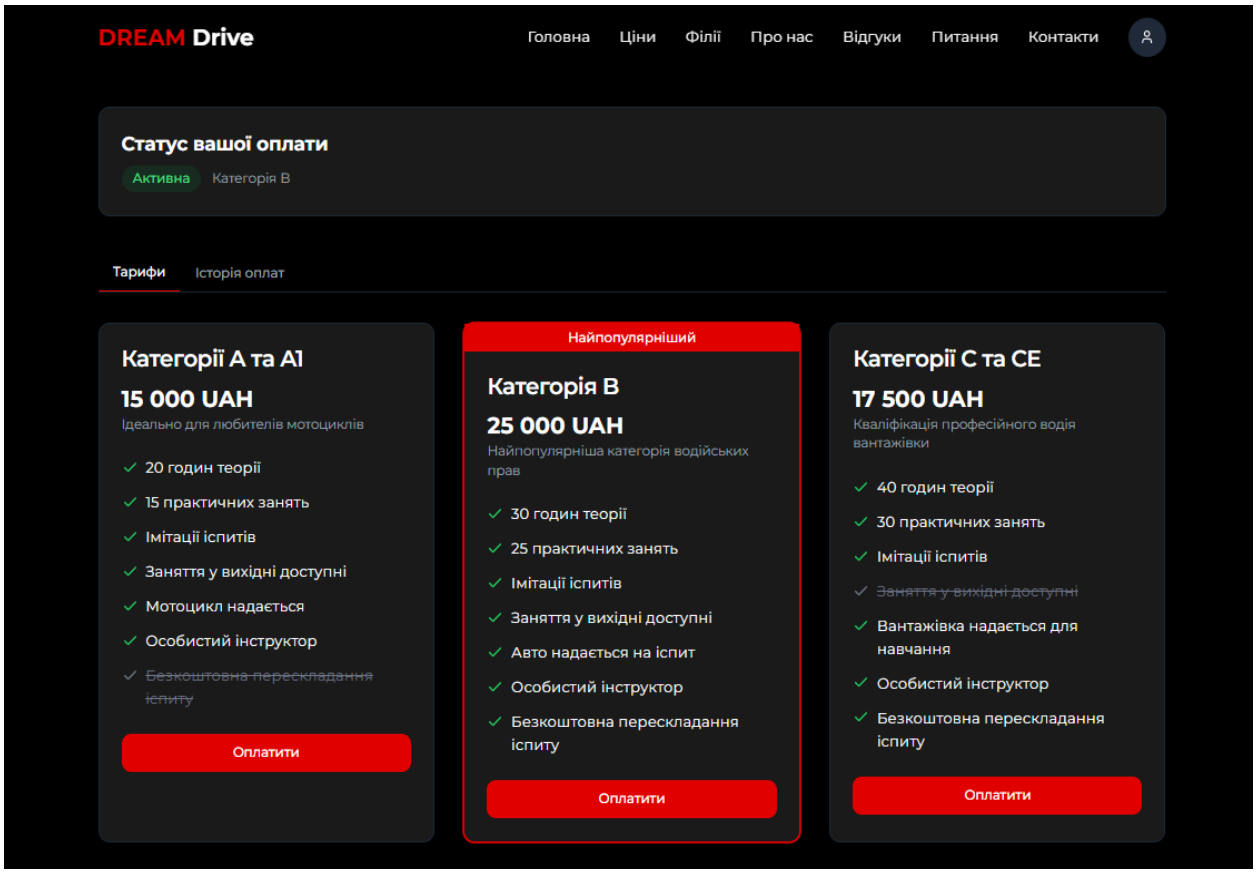


Рисунок 4.6 – Прототип сторінки платежів студента

– сторінка перегляду та бронювання занять студентом (рисунок 4.7);

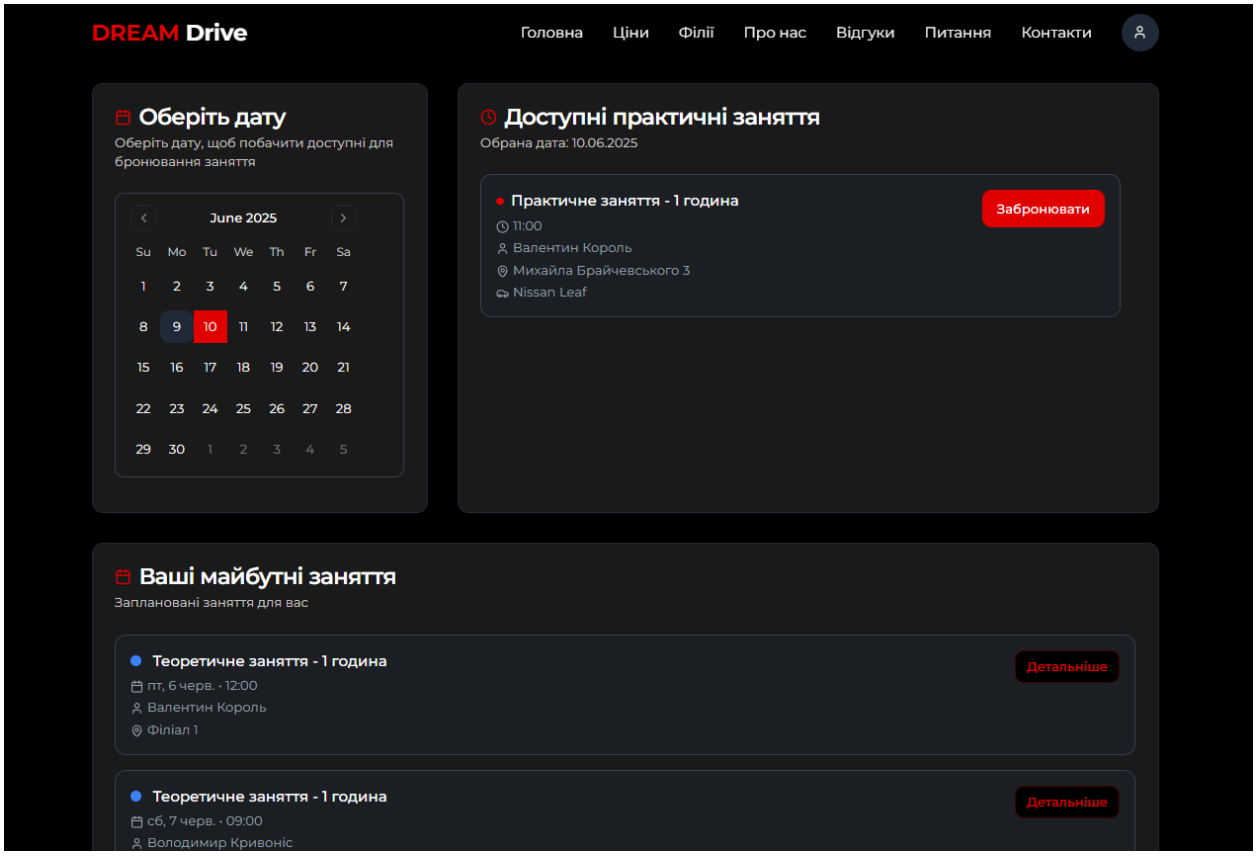


Рисунок 4.7 – Прототип сторінки перегляду та бронювання занять студентом

– сторінка перегляду навчальних матеріалів (рисунок 4.8);

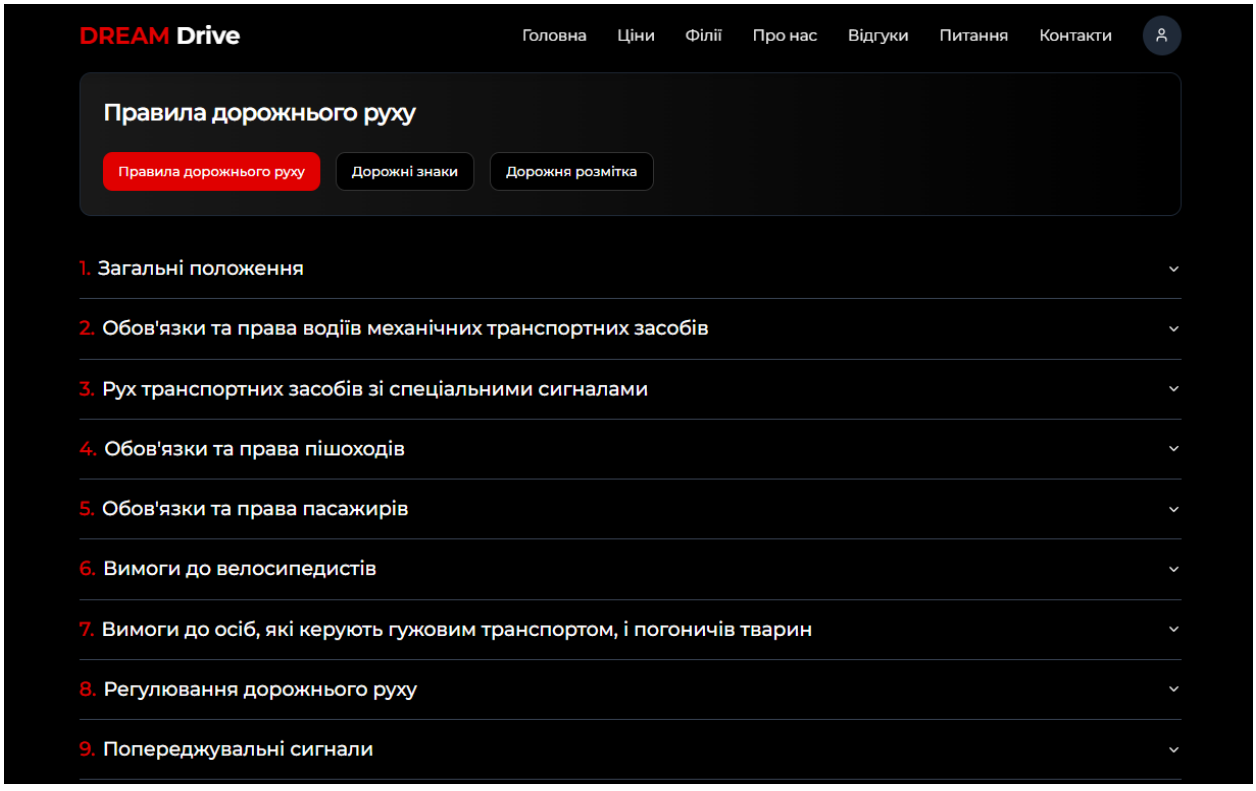


Рисунок 4.8 – Прототип сторінки перегляду навчальних матеріалів

– сторінка з тестами для студентів (рисунок 4.9);

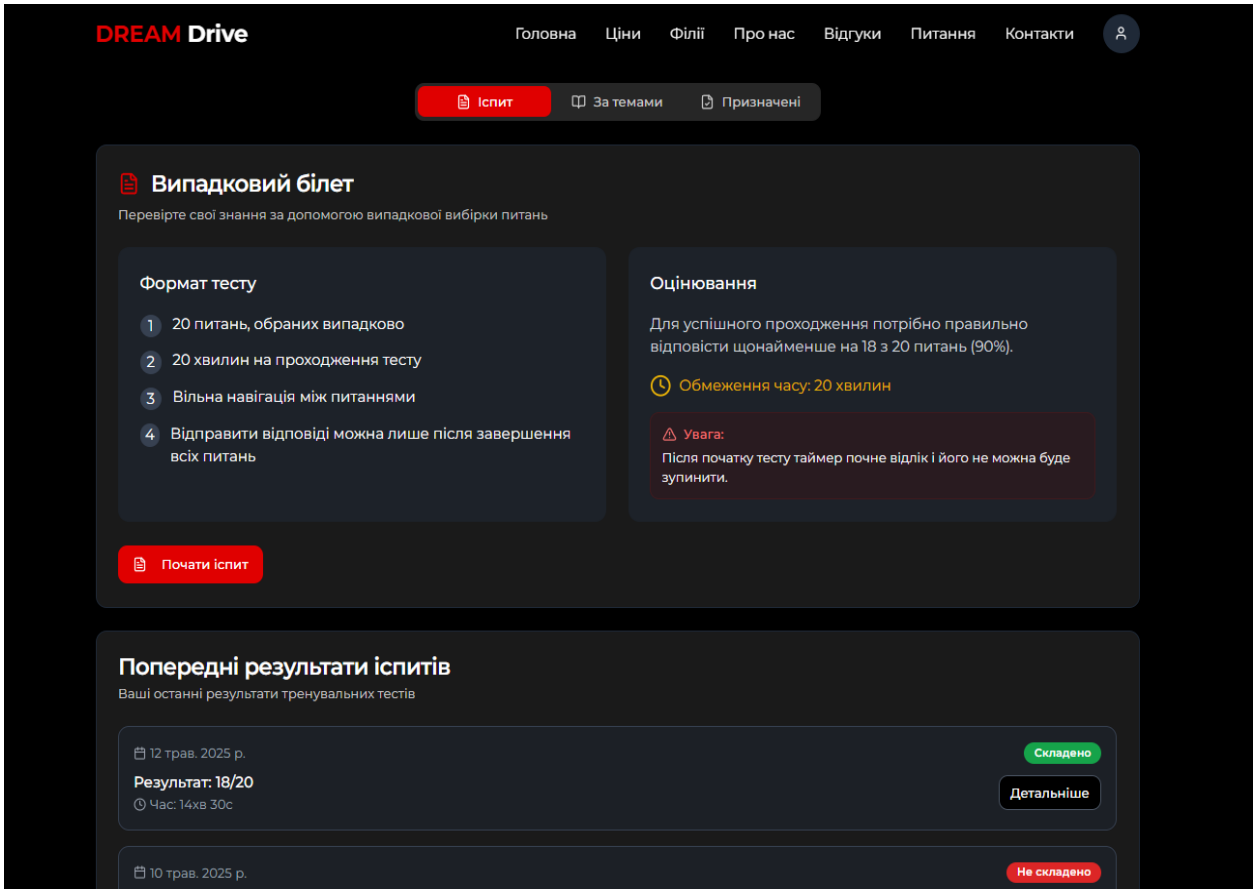


Рисунок 4.9 – Прототип сторінки з тестами для студентів

– сторінка проходження тесту (рисунок 4.10);

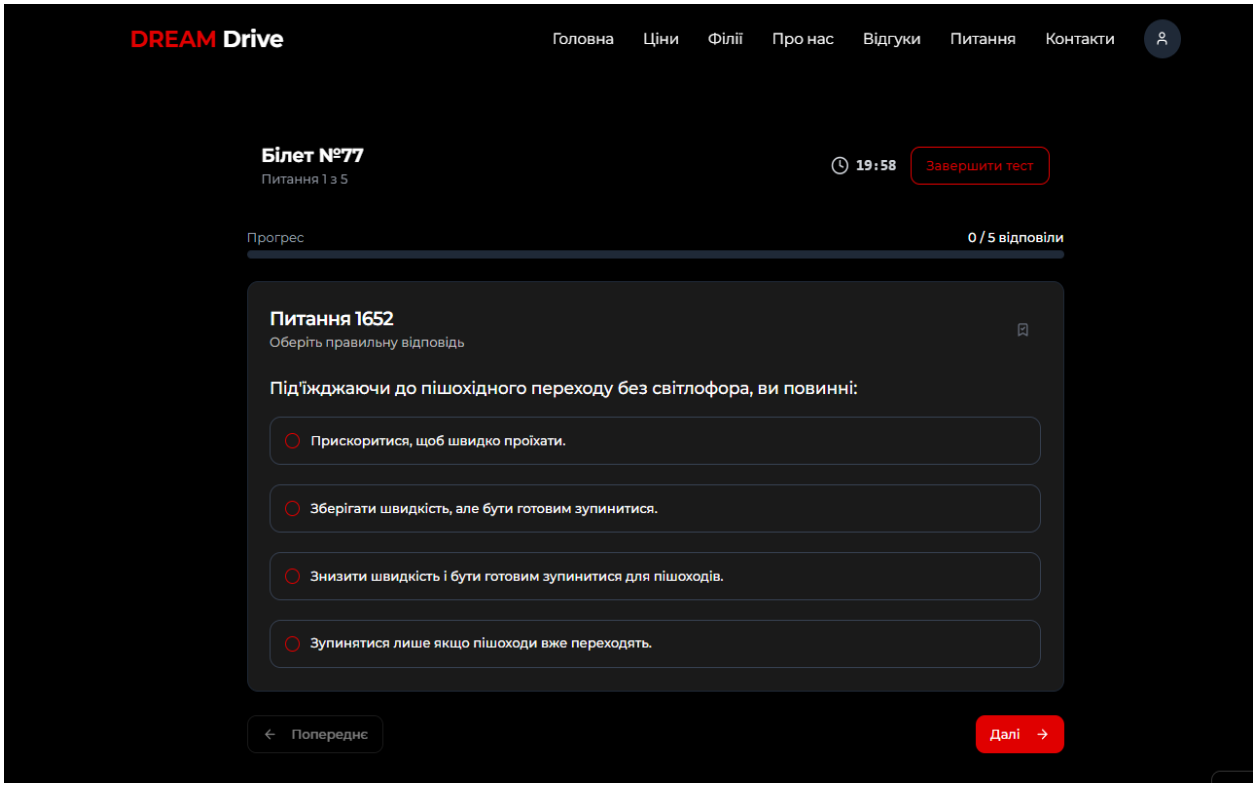


Рисунок 4.10 – Прототип сторінки проходження тесту

– сторінка результату тесту (рисунок 4.11);

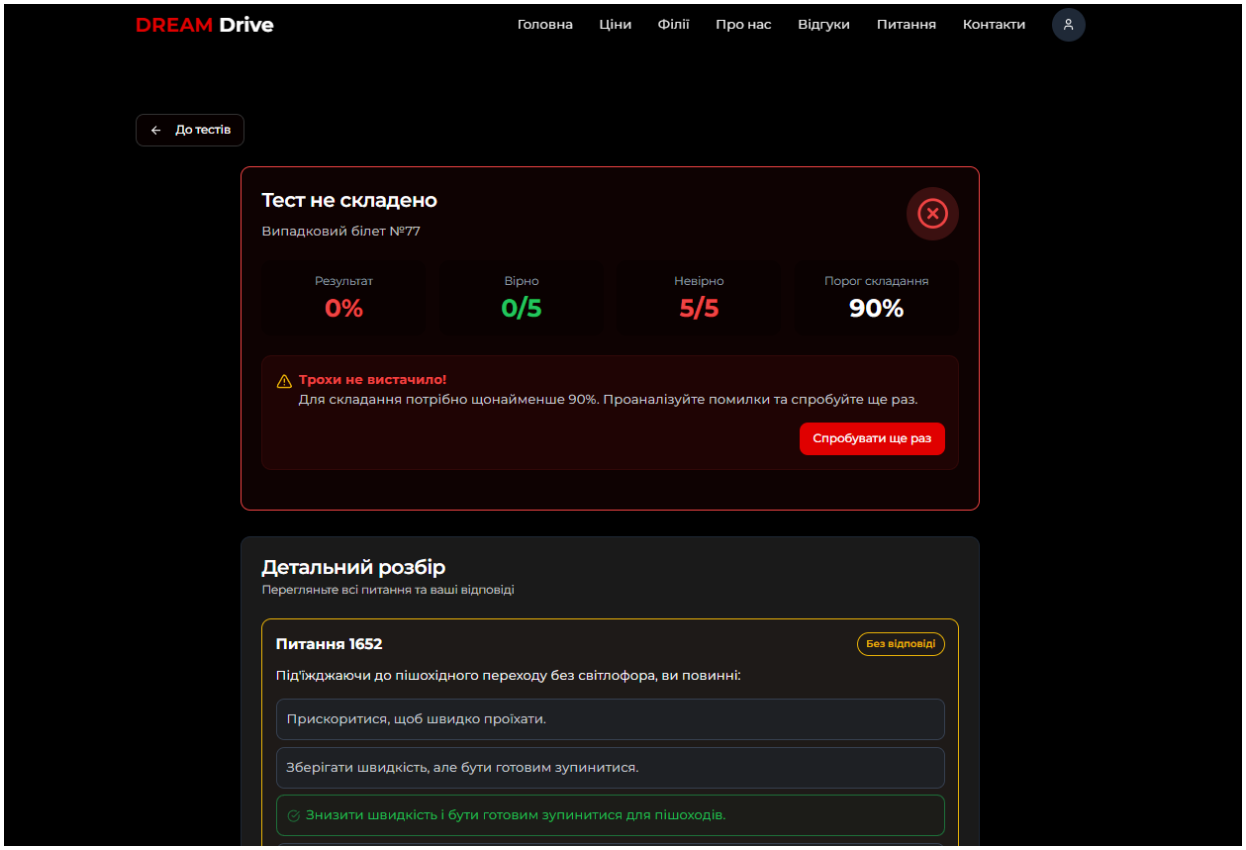


Рисунок 4.11 – Прототип сторінки результату тесту

– сторінка панелі керування викладача (рисунок 4.12);

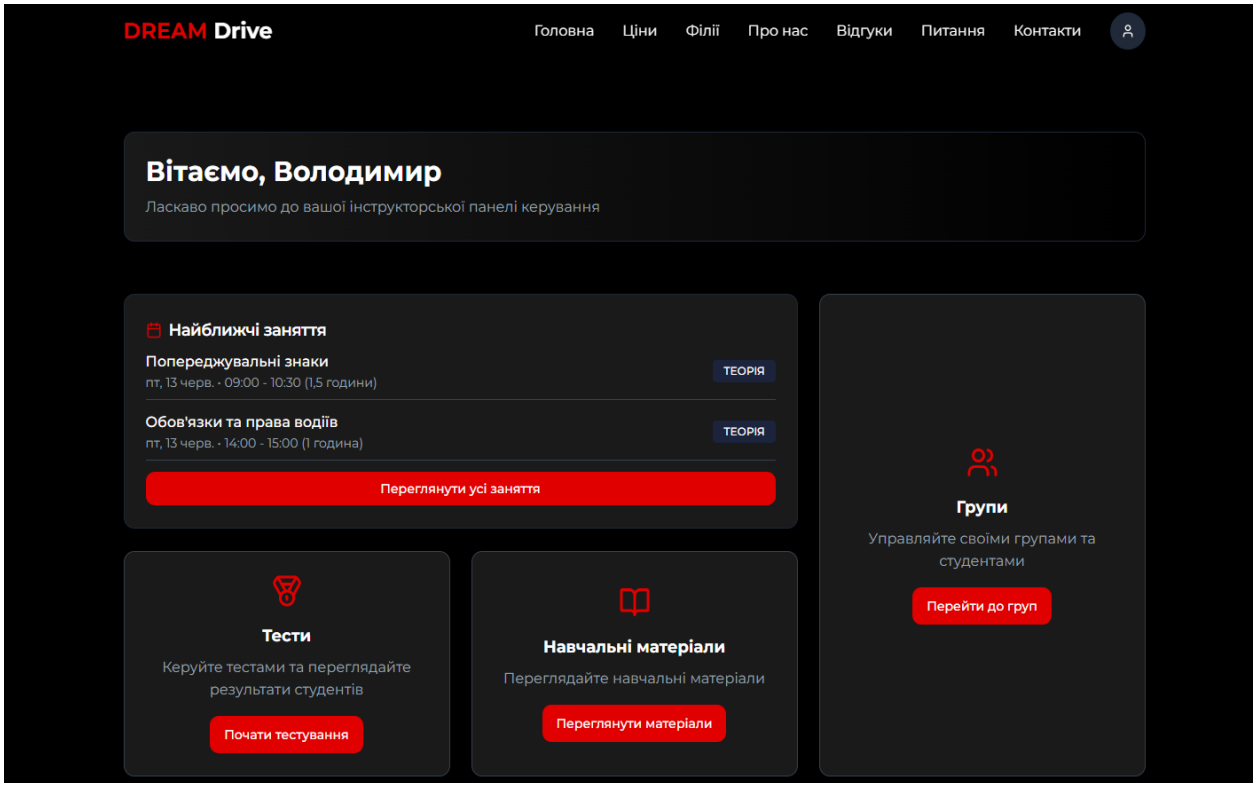


Рисунок 4.12 – Прототип сторінки панелі керування викладача

– сторінка перегляду груп та студентів (рисунок 4.13);

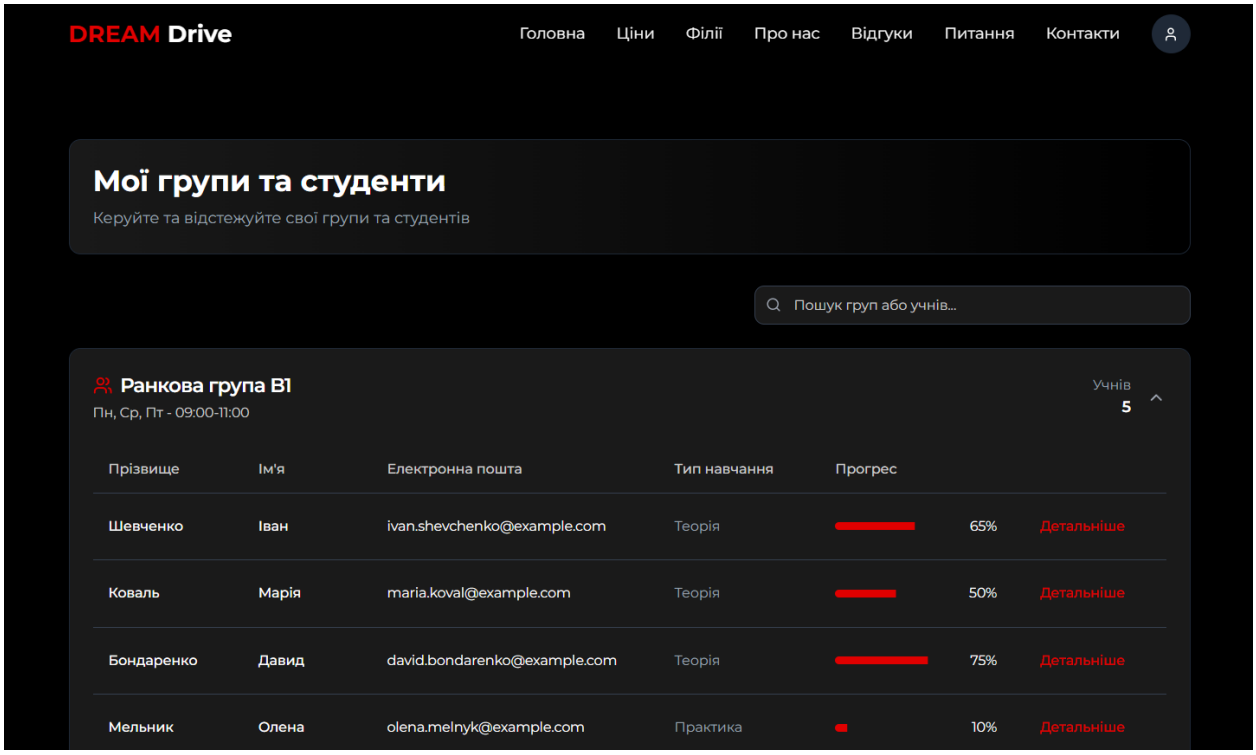


Рисунок 4.13 – Прототип сторінки перегляду груп та студентів

– сторінка управління заняттями за допомогою інтерактивного календаря викладача (рисунок 4.14);

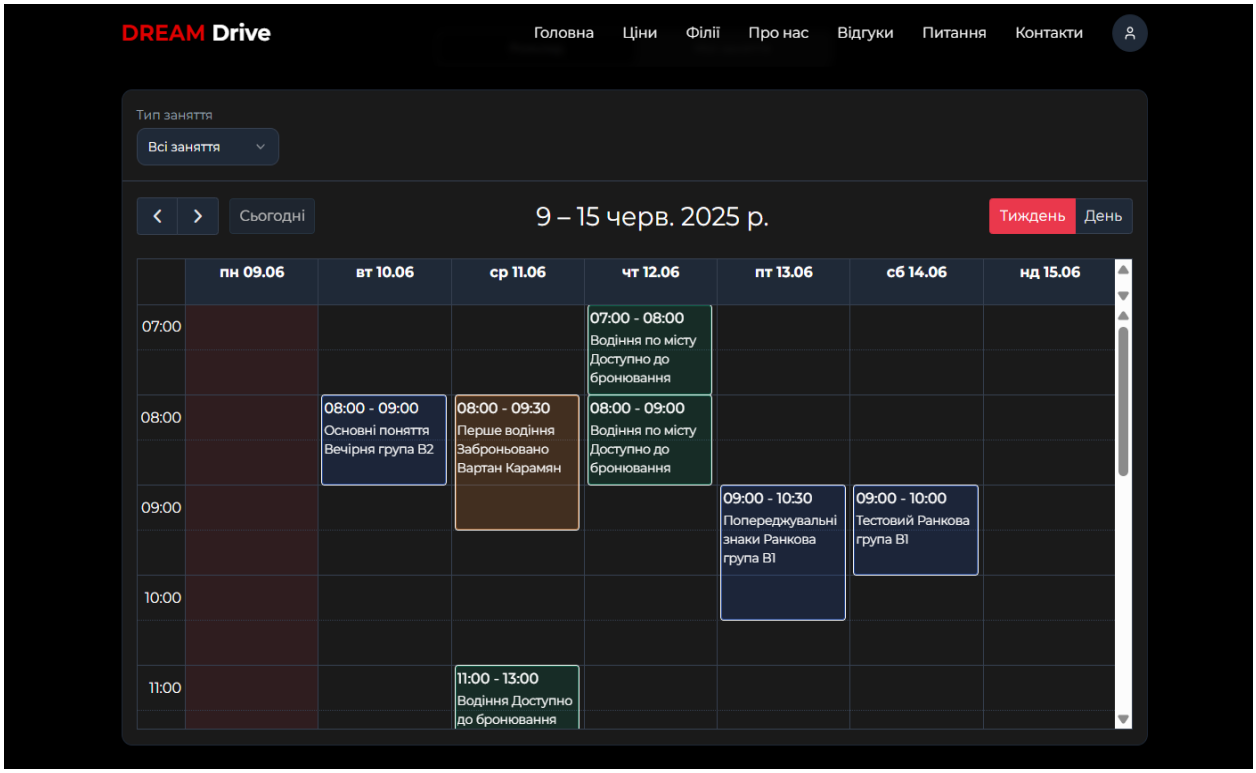


Рисунок 4.14 – Прототип сторінки управління заняттями за допомогою інтерактивного календаря викладача

– сторінка управління тестами для викладачів (рисунок 4.15);

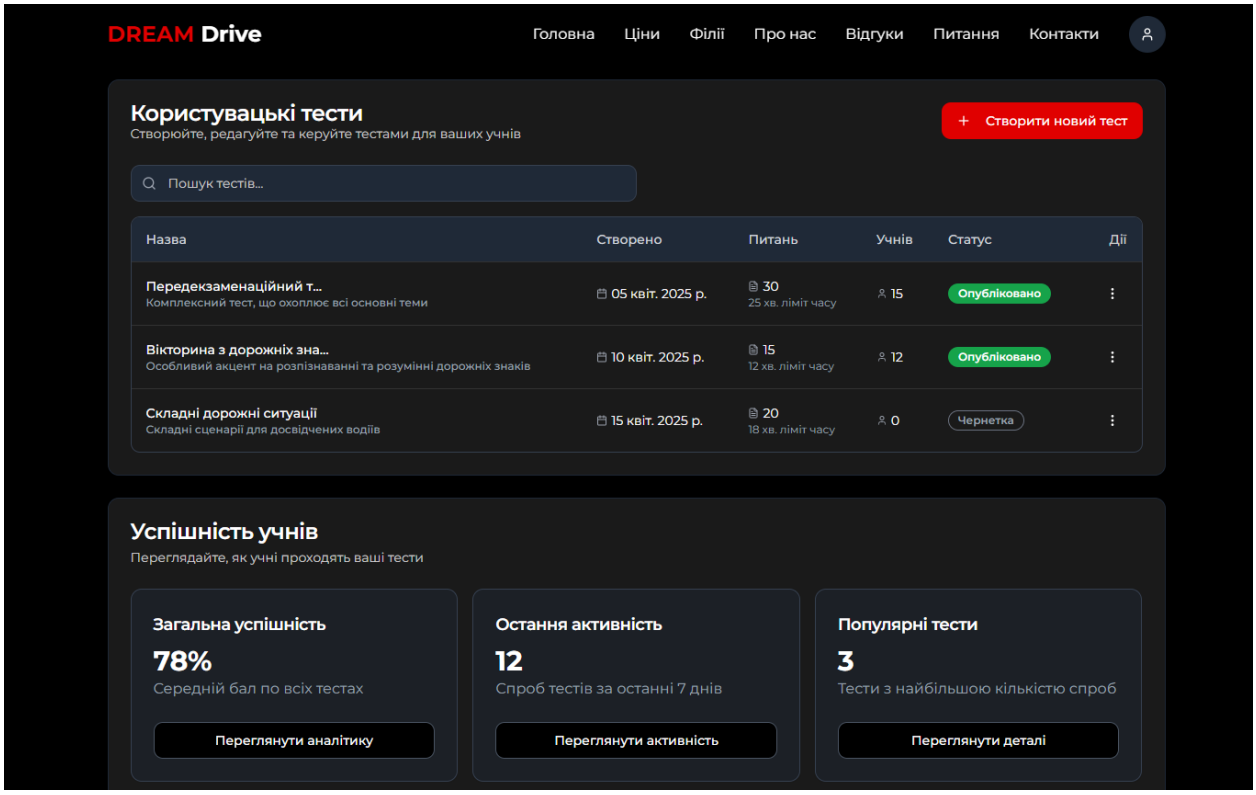


Рисунок 4.15 – Прототип сторінки управління тестами для викладачів

– сторінка створення нового тесту (рисунок 4.16);

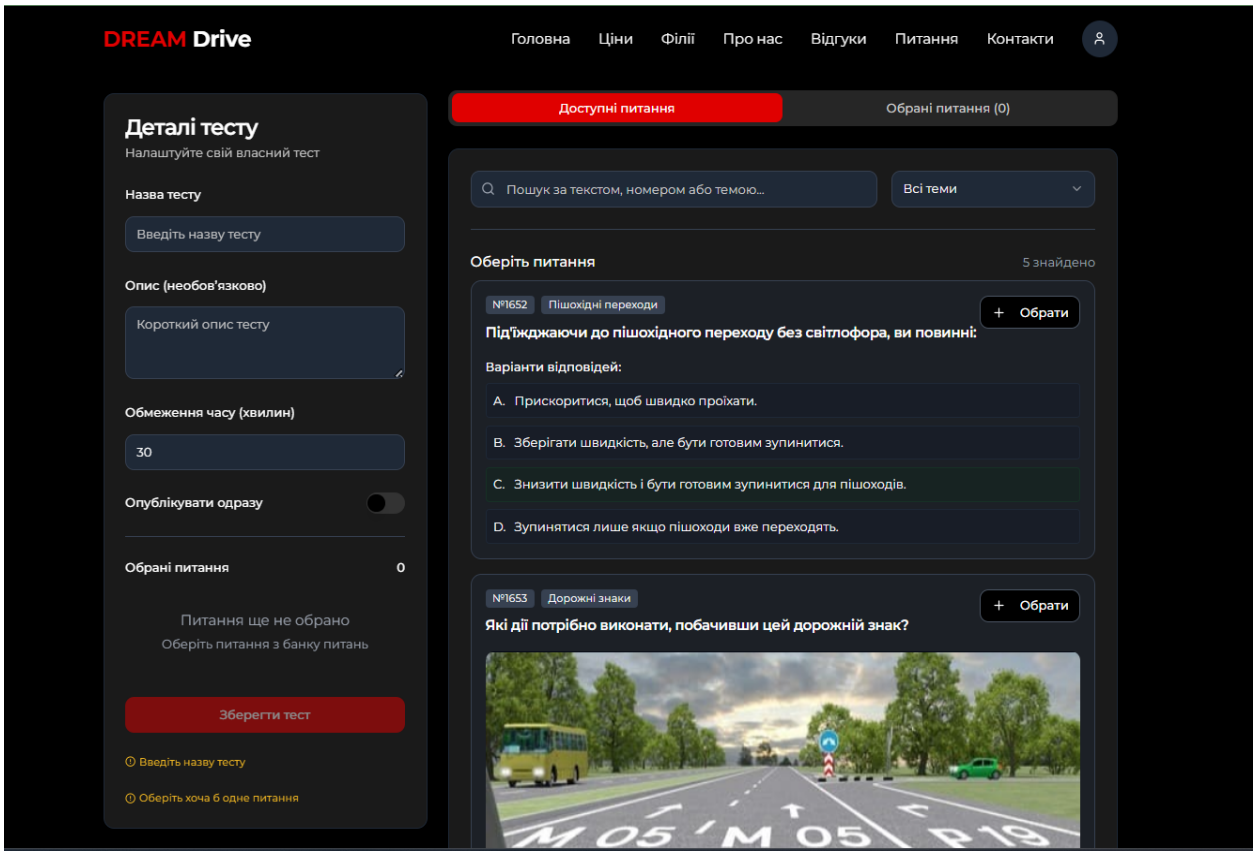


Рисунок 4.16 – Прототип сторінки створення нового тесту

– сторінка панелі керування адміністратора (рисунок 4.17);

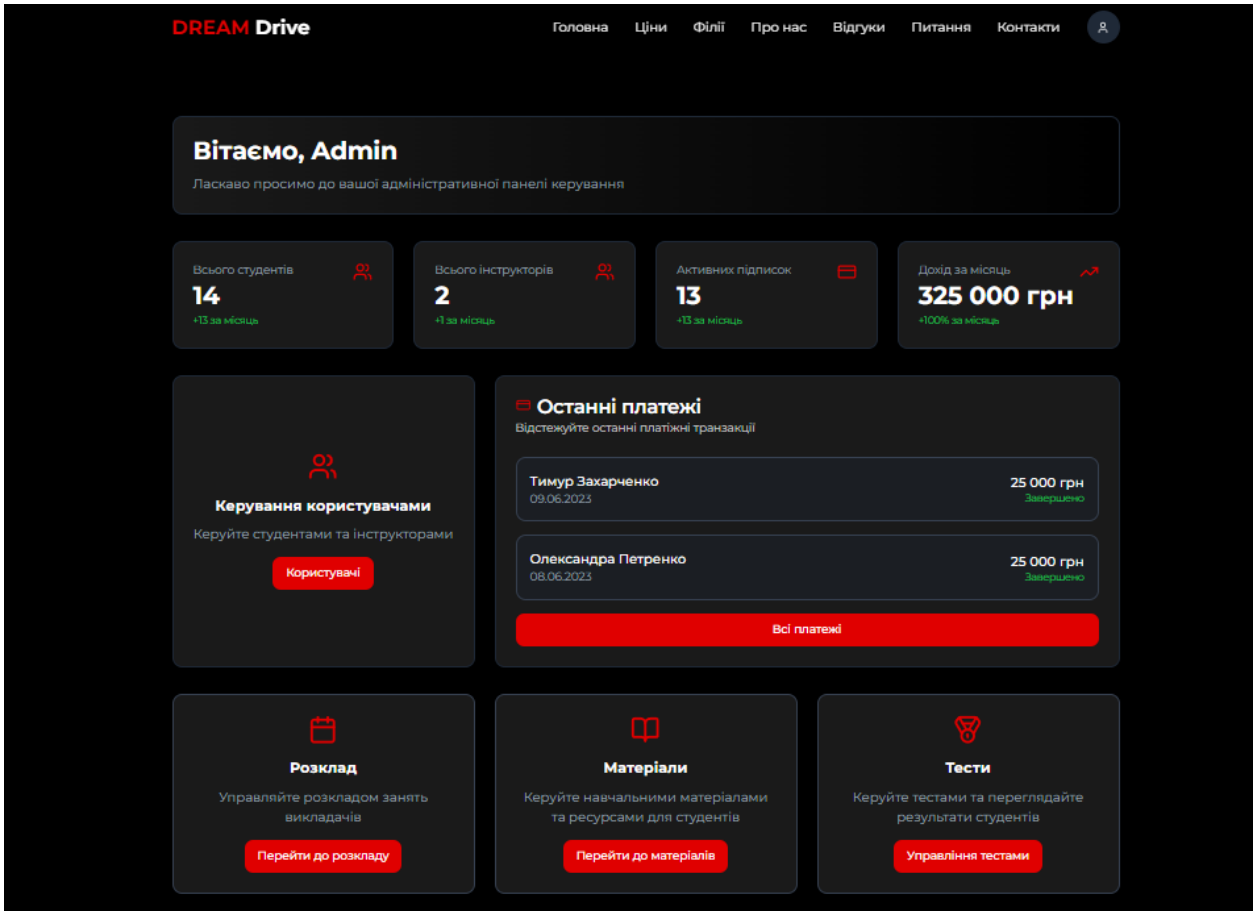


Рисунок 4.17 – Прототип сторінки панелі керування адміністратора

– сторінка для керування користувачами (рисунок 4.18);

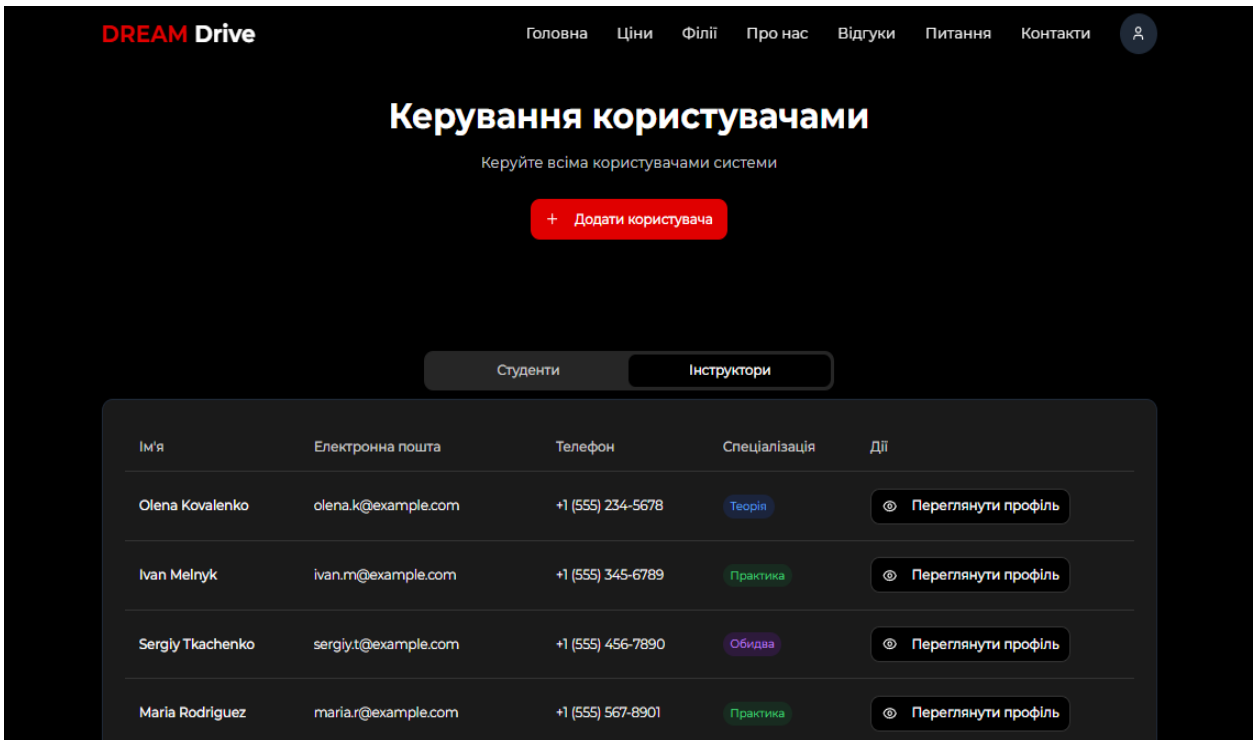


Рисунок 4.18 – Прототип сторінки для керування користувачами

– сторінка для огляду платежів (рисунок 4.19);

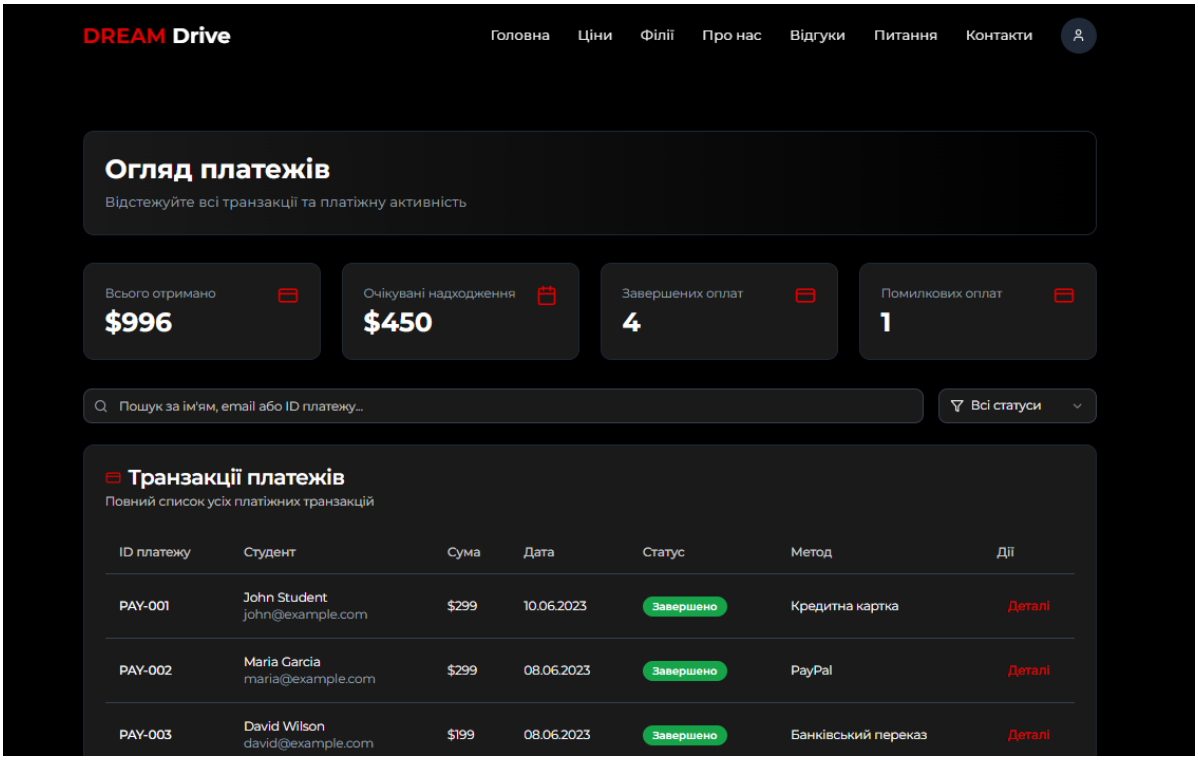


Рисунок 4.19 – Прототип сторінки для огляду платежів

– сторінка управління розкладом за допомогою інтерактивного календаря адміністратора (рисунок 4.20);

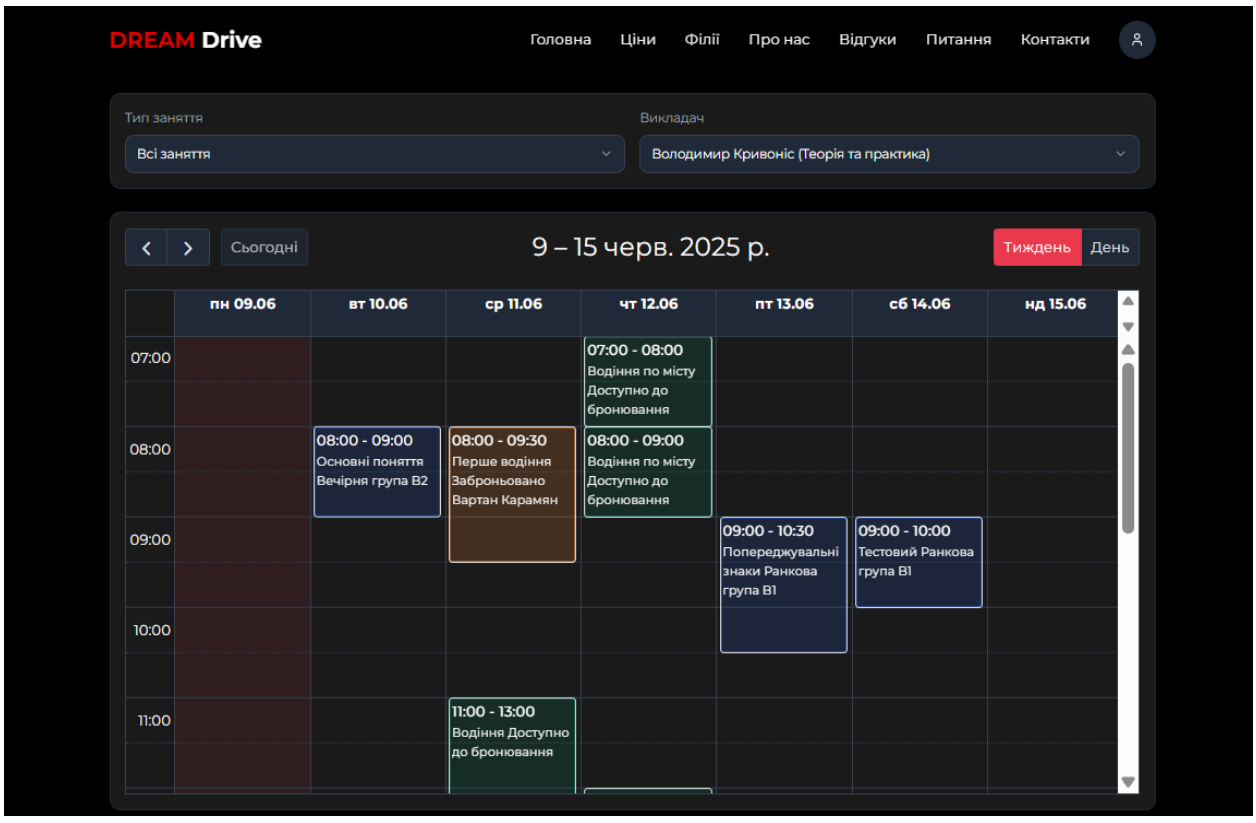


Рисунок 4.20 – Прототип сторінки управління розкладом

– сторінка управління навчальними матеріалами для адміністратора (рисунок 4.21);

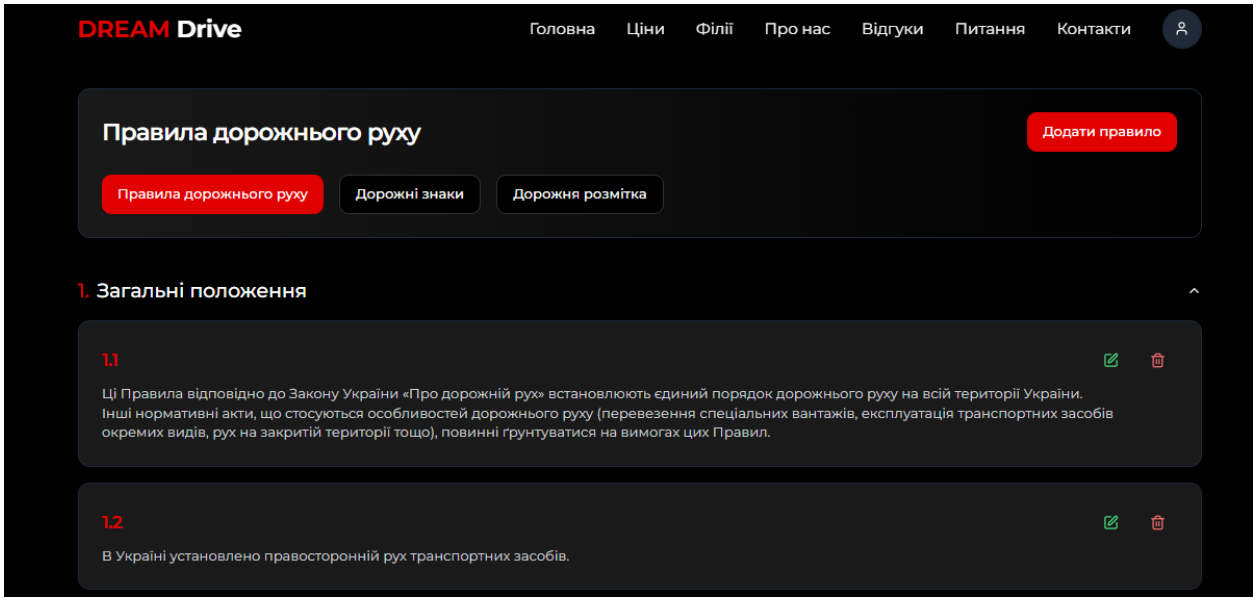


Рисунок 4.21 – Прототип сторінки управління навчальними матеріалами для адміністратора

– сторінка управління відгуками для адміністратора (рисунок 4.22).

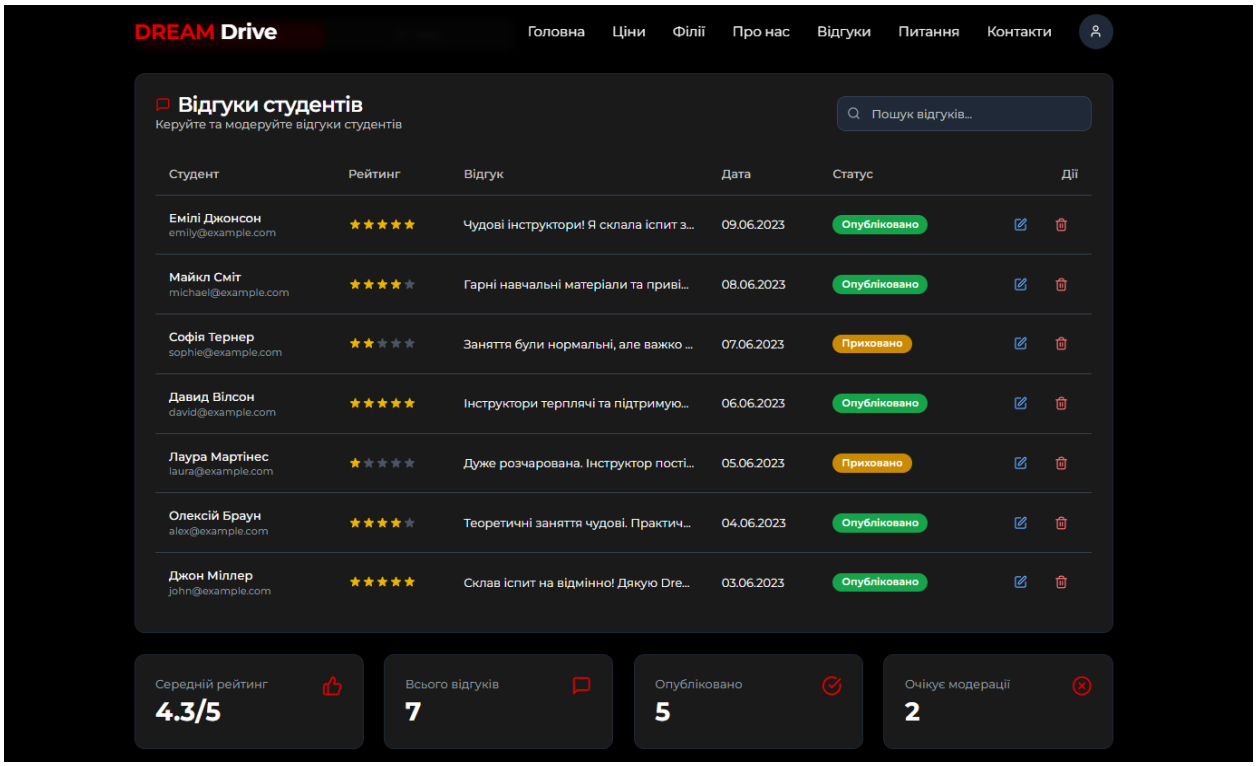


Рисунок 4.22 – Прототип сторінки управління відгуками для адміністратора

4.1.2 Головна сторінка:

4.1.2.1 Для неавторизованого користувача:

– перегляд головної сторінки з описом автошколи;
– перегляд інформації про ціни, філії, відгуки, FAQ та контакти;
– доступ до форми зворотного зв’язку;
– можливість переходу до реєстрації або входу в систему.

4.1.2.2 Для адміністратора:

– керування переліком цінових планів;
– перегляд та публікація на головній сторінці відгуків користувачів;
– редагування розділів FAQ .

4.1.2.3 Додаткові вимоги:

– для ролей студент та викладач вимогі такі ж, як і для неавторизованого користувача;
– головна сторінка має бути адаптивною (підтримка перегляду з мобільних пристроїв).

4.1.3 Авторизація та управління користувачами:

4.1.3.1 Для неавторизованого користувача:

– реєстрація нового облікового запису;
– вхід у систему з використанням email та пароля;
– авторизація через Google-акаунт;
– відновлення пароля через email із підтвердженням.

4.1.3.2 Для авторизованого користувача:

– перегляд власного профілю;
– редагування особистої інформації;
– зміна пароля через профіль;
– вихід із системи.

4.1.3.3 Для викладача:

– перегляд списку навчальних груп, до яких він прикріплений;

– перегляд списку студентів у кожній групі.

4.1.3.4 Для адміністратора:

– створення користувачів;

– редагування профілю будь-якого користувача;

– перегляд і пошук усіх користувачів у системі;

– видалення облікових записів.

4.1.3.5 Додаткові вимоги:

– email повинен бути обов’язковим, відповідати стандартному формату та бути унікальним серед усіх зареєстрованих користувачів;

– пароль має відповідати вимогам безпеки: мінімальна довжина не менше 8 символів; пароль не повинен бути занадто схожим на персональні дані користувача; не може бути загальновживаним (наприклад, “123456”, “qwerty”, “password”); не повинен складатися лише з цифр;

– пароль повинен бути введений двічі (password1, password2), і обидва введення мають збігатися.

4.1.4 Платежі:

4.1.4.1 Для неавторизованого користувача:

– перегляд інформації про вартість навчання на публічній сторінці.

4.1.4.2 Для студента:

– здійснення онлайн-оплати через інтеграцію з LiqPay;

– оновлення доступу після успішної оплати;

– перегляд власної історії оплат із зазначенням дати, суми та статусу транзакції.

4.1.4.3 Для адміністратора:

– перегляд усіх транзакцій системи, відфільтрованих за датою, філією, студентом або статусом;

– перевірка статусу оплати для кожного студента.

4.1.4.4 Додаткові вимоги:

– інтеграція з платіжним сервісом LiqPay через захищений API;

– успішна оплата повинна автоматично активувати доступ до навчального функціоналу;

– користувач має отримувати повідомлення про результат транзакції (успіх або помилка).

4.1.5 Навчальні матеріали:

4.1.5.1 Для неавторизованого користувача:

– перегляд загального опису тем або прикладів навчальних матеріалів (у вигляді демо-доступу);

– перехід до реєстрації для отримання повного доступу до навчального контенту;

4.1.5.2 Для студента та викладача:

– перегляд навчальних матеріалів з ПДР, включаючи текст та зображення;

– навчальні матеріали структуровані за темами та розділами;

4.1.5.3 Для адміністратора:

– створення, редагування та видалення навчальних матеріалів;

4.1.6 Розклад занять:

4.1.6.1 Для неавторизованого користувача:

– перегляд загальної інформації про навчальний процес автошколи;

– відсутність доступу до розкладу, з пропозицією авторизуватись або зареєструватися для повного доступу.

4.1.6.2 Для студента:

– перегляд списку власних запланованих занять із зазначенням та статусу;

– бронювання доступних практичних занять з урахуванням вільних слотів викладача;

– отримання повідомлення, якщо вже заброньовано одне заняття й нове бронювання неможливе;

– перегляд оновленого розкладу після підтвердження бронювання.

4.1.6.3 Для викладача:

– перегляд запланованих занять у вигляді списку;

– перегляд особистого розкладу у вигляді подій у календарі;

– додавання нових занять через натискання на вільний таймслот і заповнення форми;

– редагування або скасування заняття, якщо до його початку залишилось щонайменше 24 години.

4.1.6.4 Для адміністратора:

– перегляд розкладу викладачів у вигляді подій у календарі;

– додавання, редагування та скасування занять, аналогічно викладачеві.

4.1.6.5 Додаткові вимоги:

– всі календарні операції мають супроводжуватись спливаючими повідомленнями про статус операції;

– забезпечення валідації при створенні/редагуванні занять (уникнення накладання занять, перевірка вільного часу викладача);

– зміна масштабу календаря (день/тиждень);

– інтерфейс календаря має підтримувати фільтрацію (тип заняття, викладач, дата).

4.1.7 Тестування:

4.1.7.1 Для неавторизованого користувача:

– відсутність доступу до тесування, з пропозицією авторизуватись або зареєструватися для повного доступу.

4.1.7.2 Для студента:

– можливість вибору типу тесту: білети (як на іспиті), питання по темах або тест, призначений викладачем;

– проходження тесту з відображенням прогресу, таймера, навігації між питаннями;

– отримання результату тесту одразу після проходження: оцінка, кількість правильних відповідей, час виконання;

– можливість перегляду помилок з поясненнями для неправильних відповідей;

– доступ до історії пройдених тестів у секції «Результати тестів» із зазначенням типу тесту, дати проходження, оцінки та статусу;

– можливість відкриття конкретного тесту для перегляду відповідей та правильних варіантів.

4.1.7.3 Для викладача:

– доступ до модуля створення тестів;

– можливість створення нового тесту: введення назви, опису, вибір кількості запитань;

– можливість вибору питань з офіційного пулу (з фільтрацією за темами);

– перегляд результатів студентів по кожному тесту: ім’я студента, дата проходження, результат.

4.1.7.4 Додаткові вимоги:

– автоматичне завершення тесту після останнього питання або завершення часу;

– повідомлення у випадку технічної помилки або недоступності тесту з рекомендацією повторити спробу пізніше;

– забезпечення валідації при створенні тестів (мінімальна кількість питань, унікальність назв);

– супровід усіх основних дій спливаючими повідомленнями про статус (успішно / помилка).

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити достовірність та узгодженість даних під час запису, оновлення або видалення інформації в базі даних. Захист даних має включати обмеження доступу відповідно до ролей користувачів, валідацію вхідних даних на стороні клієнта та сервера, а також запобігання основним типам атак, зокрема SQL-ін’єкціям. Усі критичні дії з даними повинні супроводжуватись логуванням для подальшого аудиту.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 8 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 16 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення має бути сумісним з актуальними версіями популярних веб-браузерів (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) та підтримувати роботу як на настільних, так і на мобільних пристроях. Серверна частина вебзастосунку повинна коректно функціонувати в середовищах, що базуються на сучасних версіях операційних систем Linux або Windows Server.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: png.

4.5.2 Вимоги до вихідних даних

Вихідні дані відсутні

4.5.3 Вимоги до мови розробки

Розробку виконати на мовах програмування Python та TypeScript.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформах Visual Studio Code та PyCharm Community.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді GitHub репозиторію.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

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

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

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

– схема структурна варіантів використання;
– схема бази даних;
– архітектура програмного забезпечення;
– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

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

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проєкту	15.03	
2.	Розробка технічного завдання	29.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	12.04	Специфікації програмного забезпечення
4.	Проєктування структури програмного забезпечення, проєктування компонентів	19.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	20.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	25.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проєкту	27.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проєкту	01.06	Графічний матеріал проєкту
9.	Оформлення технічної документації проєкту	04.06	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

Пояснювальна записка
до дипломного проєкту

на тему: Вебзастосунок для організації роботи автошколи

КПІ.ІП-1313.045440.02.81

Київ – 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CRM	–	Customer Relationship Management – система управління взаємовідносинами з клієнтами
СУБД	–	Система управління базами даних
ПДР	–	Правила дорожнього руху
JSON	–	JavaScript Object Notation – Об’єктна нотація JavaScript
JWT	–	JSON Web Token
DRF	–	Django REST Framework
IDE	–	Integrated Development Environment – інтегроване середовище розробки
API	–	Application Programming Interface – прикладний програмний Інтерфейс
ER	–	Entity-Relation diagram
БД	–	База даних
URL	–	Uniform Resource Locator – уніфікований локатор ресурсу
XSS	–	XSS – Cross-Site Scripting – міжсайтове скриптування
SSL	–	Secure Sockets Layer – рівень захищених сокетів
CSRF	–	Cross-Site Request Forgery – міжсайтове підроблення запиту
CDN	–	Content Delivery Network – мережа доставки контенту

ВСТУП

В епоху розвитку інформаційних технологій дедалі більше навчальних закладів, зокрема й автошкіл, потребують автоматизації внутрішніх процесів. З іншої сторони, сучасні користувачі очікують не лише якісного навчання, а й зручного онлайн-сервісу: доступу до матеріалів і тестів у будь-який час, онлайн навчання та оплата, керування розкладом. Водночас як викладачам, так і адміністрації потрібні інструменти для організації занять, моніторингу навчального процесу, керування навантаженням і взаємодії з учнями в єдиному цифровому середовищі.

На ринку уже представлено рішення, що частково покривають потреби автошкіл: сервіси для вивчення ПДР та проходження тестів, окремі CRM-системи або прості вебдодатки для привернення уваги клієнту, що дозволяють переглянути інформацію про послуги та залишити заявку. Однак, ці інструменти не інтегровані між собою та охоплюють лише окремі аспекти навчального процесу, що обмежує їх ефективність та зручність. Відсутність єдиної платформи, що охоплювала б усі ключові процеси — від залучення клієнтів до навчання і адміністрування — є проблемою, яку відчувають як власники автошкіл, так і їх учні та викладачі.

У рамках дипломного проєкту буде створено вебзастосунок для організації роботи автошколи. Система надаватиме інструменти для залучення студентів, прийому оплат, проведення навчання й тестування, а також для ефективного керування розкладом і обліку студентів. Метою проєкту є підвищення ефективності управління освітньою діяльністю автошколи шляхом консолідації кількох розрізнених цифрових рішень в єдину інтегровану платформу. Очікується скорочення кількості окремих застосунків з трьох до одного, що дозволить зменшити адміністративне навантаження, прискорити обробку інформації та покращити взаємодію між усіма учасниками освітнього процесу. Розроблений застосунок може бути впроваджений як в окремих автошколах, так і в мережах навчальних центрів.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

Дипломне проєктування передбачає виконання наступних завдань:

– аналіз предметної області та опис її ключових бізнес-процесів, визначення загального завдання розробки у рамках ДП;

– аналіз існуючих рішень (як програмних продуктів, так і алгоритмічних чи технічних рішень) обраного завдання розробки у рамках ДП;

– аналіз та моделювання бізнес-процесів;

– розроблення функціональних, нефункціональних та системних вимог до програмного забезпечення;

– аналіз економічних показників програмного забезпечення ДП;

– постановка завдання на розробку програмного забезпечення ДП;

– розроблення архітектури програмного забезпечення;

– розроблення архітектурних рішень та обґрунтування вибору засобів розробки програмного забезпечення;

– конструювання та розроблення програмного забезпечення;

– аналіз безпеки даних програмного забезпечення;

– аналіз якості та тестування програмного забезпечення;

– розгортання та супровід програмного забезпечення;

– створення супроводжувальної документації до розробленого програмного забезпечення.

1.2 Аналіз предметної області

Автошкола — це навчальний заклад, який спеціалізується на підготовці майбутніх водіїв транспортних засобів. Головним завданням таких установ є комплексне навчання, що включає засвоєння теоретичних знань правил дорожнього руху (ПДР) та формування практичних навичок керування автомобілем, а також підготовку учнів до складання іспитів державного рівня

у сервісних центрах Міністерства внутрішніх справ України [1]. Освітній процес в автошколі зазвичай поділяється на кілька етапів: теоретична підготовка, практичні заняття з водіння, проміжне оцінювання знань, адміністративне супроводження учнів та координація подальшого навчання.

Спочатку проходить теоретична частина навчання, яка включає вивчення правил дорожнього руху, дорожніх знаків, основ безпеки руху, правил експлуатації транспортних засобів, а також роз'яснення норм законодавства, що регулює дорожній рух. Після успішного складання теоретичного іспиту у сервісному центрі, учні переходять до безпосередньо практики. Практичні заняття спрямовані на розвиток навичок керування транспортним засобом у різних дорожніх умовах, з урахуванням технічних особливостей автомобіля, а також підготовку до реальних ситуацій на дорозі. Ці заняття проходять під керівництвом досвідчених інструкторів, які відповідають за безпеку учнів і ефективність навчального процесу [2].

Для успішної організації навчання автошкола повинна забезпечувати ефективну координацію між усіма учасниками процесу: учнями, викладачами теоретичної та практичної частин, адміністрацією, що відповідає за реєстрацію, формування розкладу, контроль оплати та документообіг. Цей комплекс взаємодій формує низку бізнес-процесів, які потребують системного підходу для автоматизації та оптимізації. Крім того, важливо враховувати особливості нормативно-правової бази, що регулює діяльність автошкіл, і вимоги до документації, зокрема ведення обліку учнів, звітності перед державними органами та дотримання стандартів якості освітніх послуг.

Сучасний розвиток інформаційних технологій та зміна потреб користувачів вимагають автоматизації багатьох процесів, які раніше здійснювалися вручну або з використанням окремих неінтегрованих інструментів. За даними досліджень EdTech-індустрії, серед основних напрямів розвитку навчальних сервісів виділяють створення адаптивного контенту, персоналізацію навчання, інтегровані онлайн-комунікації, аналітику успішності студентів і забезпечення доступу з мобільних пристроїв до освітніх

ресурсів [3]. Проте в сфері автошкіл більшість сучасних рішень впроваджується фрагментарно і не охоплює повний цикл навчального процесу.

На сьогодні в більшості автошкіл для організації навчання використовуються традиційні методи — записи по телефону, ведення розкладу в Excel [4] або на папері, оплата готівкою без електронних сервісів, а для вивчення ПДР часто застосовують сторонні ресурси, що не інтегровані з іншими процесами. Хоча деякі заклади мають власні сайти, вони переважно виконують інформаційну функцію, не дозволяючи комплексно управляти навчальним процесом. Через це учням доводиться взаємодіяти з кількома окремими системами, що створює незручності і знижує якість освітнього процесу.

Автошколи виконують багато бізнес-процесів, що включають реєстрацію учнів, формування розкладу занять, розподіл викладацького навантаження, проведення теоретичних і практичних уроків, організацію тестування з ПДР, оплату навчання, моніторинг успішності учнів та підтримку комунікації між учнями, викладачами і адміністрацією. Проте сучасні IT-інструменти не забезпечують цілісність і ефективність управління цими процесами, що призводить до дублювання інформації, втрати даних та обмеженості масштабування, особливо коли автошкола має кілька філій або класів. Також існують проблеми з аналізом якості навчання і недостатньою мобільністю систем, що заважає учням отримувати послуги цілодобово.

Покращення ситуації можна досягти за рахунок створення комплексних вебзастосунків із модульною архітектурою, які інтегрують всі ключові бізнес-процеси в єдину цифрову платформу, дозволяючи автоматизувати рутинні задачі, розширити аналітичні можливості та покращити загальний досвід користувачів. Саме такий шлях було обрано для реалізації в рамках дипломного проєкту DreamDrive.

DreamDrive — це вебзастосунок на основі мікросервісної архітектури, який включає сторінки з відкритим доступом, можливість авторизації для

онлайн-запису, особисті кабінети для учнів, викладачів і адміністраторів, систему формування розкладу занять, модулі для оплати, навчання ПДР і проходження тестів, а також інструменти для керування навчальним процесом. Запропоноване рішення спрямоване на цифрову трансформацію автошколи, забезпечуючи зручність, гнучкість і масштабованість під конкретні потреби навчального закладу.

1.3 Аналіз існуючих рішень

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

1.3.1 Аналіз відомих програмних продуктів

Для порівняльного аналізу було обрано чотири програмні рішення, що частково або повністю реалізують функціонал, наближений до завдань, які ставить перед собою вебзастосунок DreamDrive. Це сервіси PDR-Online [5], GreenWay [6], EasyWeek [7] та онлайн-платформа «Автошкола Час» [8]. Дані продукти охоплюють різні аспекти цифрової взаємодії у сфері підготовки водіїв — від самостійного вивчення ПДР і тестування до онлайн-запису на заняття та часткової автоматизації управління навчальними процесами. Аналіз цих рішень дозволяє оцінити рівень розвитку відповідного сегмента українського ринку та визначити конкурентні переваги власної розробки.

PDR-Online є онлайн-платформою, орієнтованою виключно на підготовку користувачів до іспитів з Правил дорожнього руху України. Сервіс надає доступ до актуальних офіційних білетів, навчальних матеріалів та інструментів для автоматичного оцінювання. Основна цінність платформи полягає в її відповідності чинній нормативній базі та простоті використання для індивідуальної підготовки. Однак PDR-Online не має функціоналу для

адміністрування автошколи чи взаємодії з викладачами, що суттєво обмежує його застосування у професійній сфері навчання водіїв (рисунок 1.1).

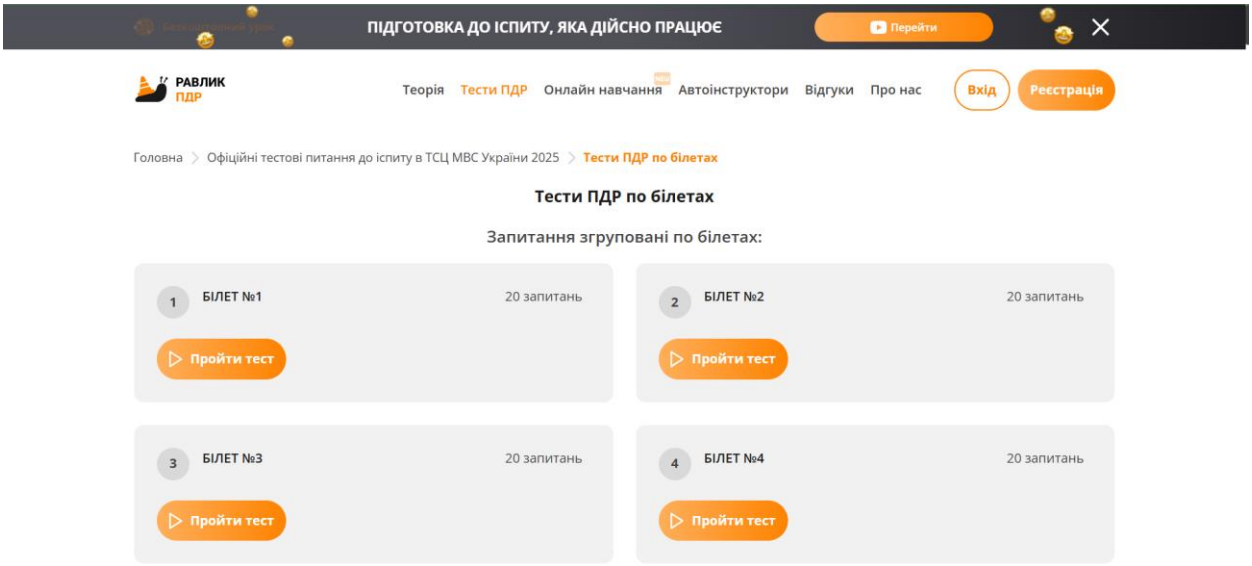


Рисунок 1.1 – Онлайн автошкола PDR-Online

Далі розглянемо ресурс для вивчення ПДР, але з дещо ширшим функціоналом – GreenWay. Окрім стандартного тестування та теоретичних матеріалів, він дозволяє викладачам формувати групи учнів, слідкувати за їхнім прогресом, організовувати тестування та здійснювати базову взаємодію в межах навчального процесу. Таким чином, GreenWay можна умовно розглядати як частково інтегровану систему керування навчанням. Проте вона не охоплює такі важливі компоненти, як планування занять, фінансовий облік, керування персоналом чи створення автошколи (рисунок 1.2).

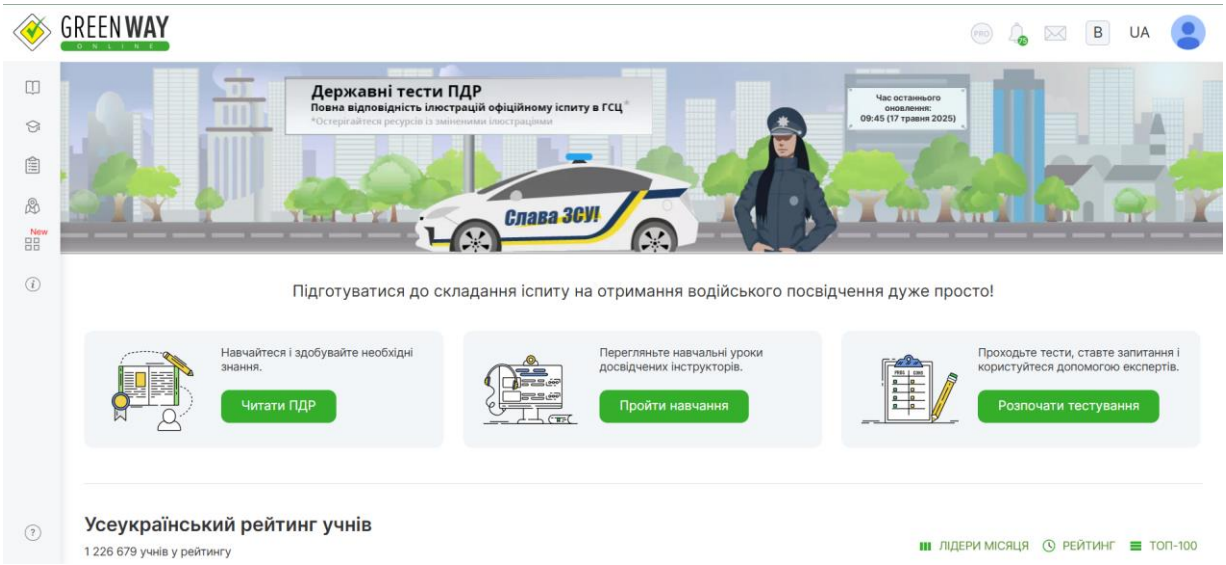


Рисунок 1.2 – Платформа GreenWay

Також є більш універсальні CRM-рішення, як наприклад, EasyWeek. Система забезпечує можливість онлайн-запису на заняття, синхронізацію з Google Calendar, управління розкладом інструкторів, автоматичне надсилання повідомлень, відстеження свого прогресу, онлайн-оплати та можливість створення сайту за допомогою вбудованого конструктора. Разом з тим EasyWeek не спеціалізується на навчальному контенті або інтеграції з ПДР-тестуванням, що потребує додаткових рішень для повноцінної організації освітнього процесу (рисунок 1.3).

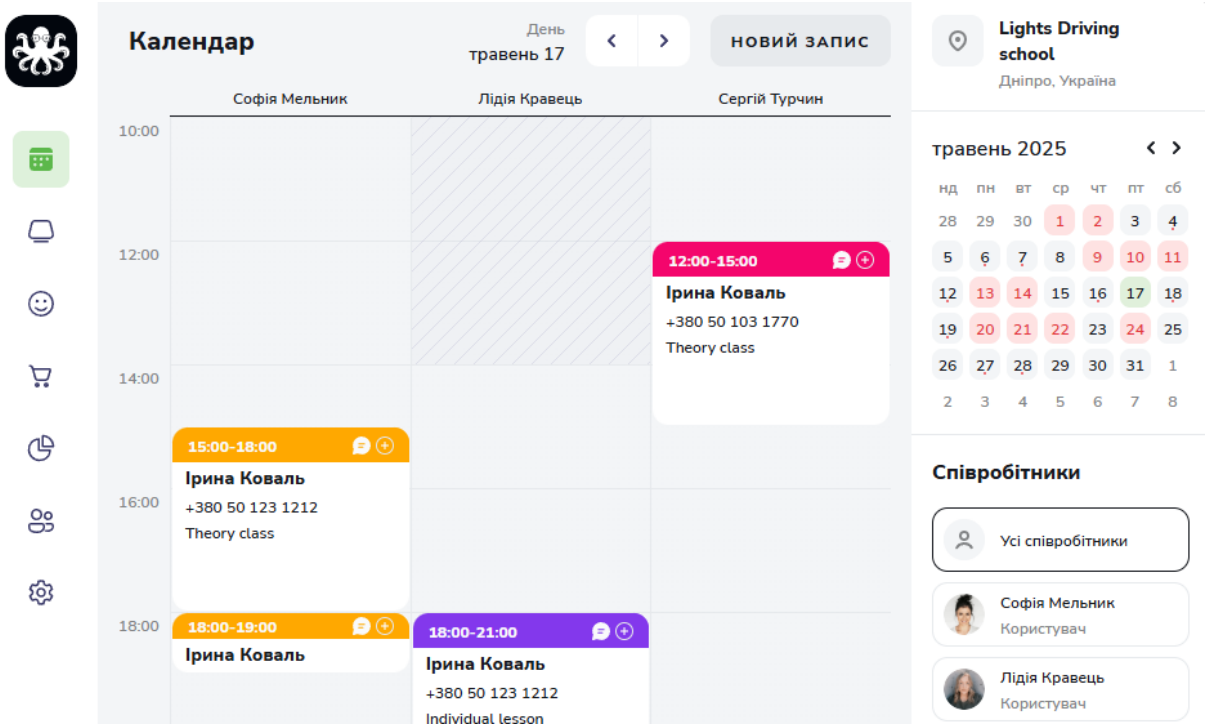


Рисунок 1.3 – CRM для автошколи EasyWeek

Онлайн-платформа «Автошкола Час» забезпечує дистанційне вивчення ПДР через особистий кабінет із відеоуроками, тематичними тестами та екзаменаційними білетами МВС, однак, попри зручний навчальний контент, вона не містить повноцінної CRM-системи: відсутні автоматизований розклад, фінансовий облік, управління персоналом та інші ключові адміністративні інструменти, тож ресурс залишається суто навчальним, не охоплюючи комплексної цифрової підтримки роботи автошколи (рисунок 1.4).

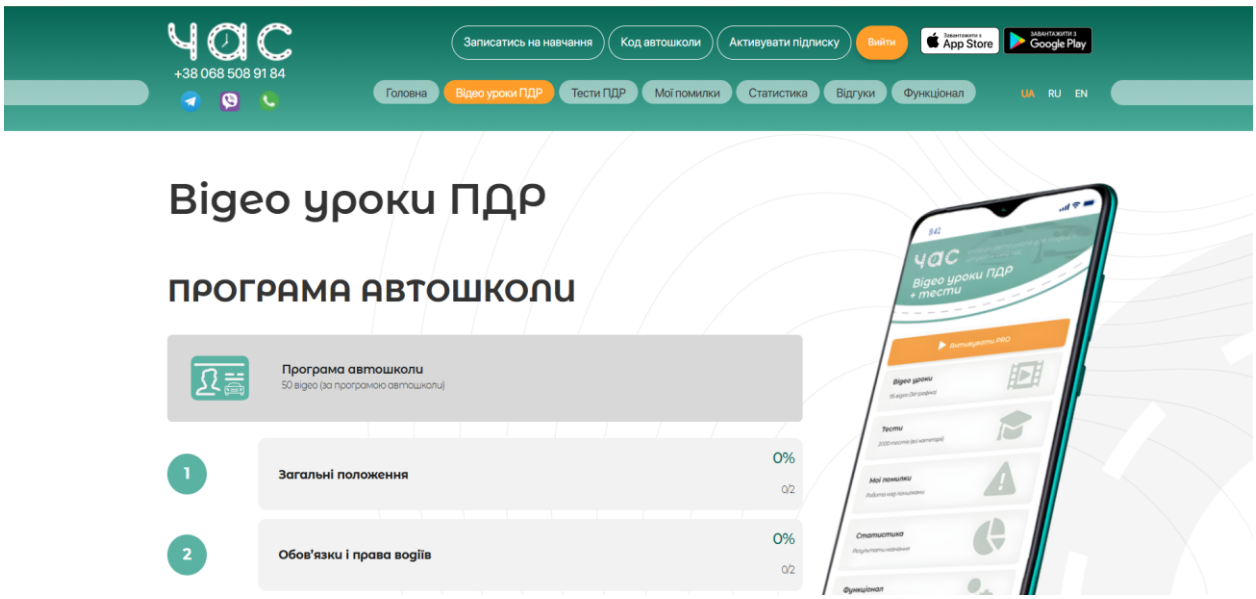


Рисунок 1.4 – «Автошкола Час»

Отже, можна зробити висновок, що жодне з рішень не поєднує в собі усіх необхідних компонентів для комплексного управління автошколою — від реєстрації учнів, організації навчального процесу, CRM-функціоналу до інтегрованого модулю підготовки до іспитів з ПДР. Це підтверджує актуальність та доцільність створення вебзастосунку DreamDrive, який має на меті об'єднати в єдиній платформі навчальні, адміністративні та аналітичні інструменти, орієнтовані на сучасні потреби автошкіл та їхніх учнів.

Для порівняння проєкту з аналогами можна скористатись таблицею 1.3.

Таблиця 1.3 – Порівняння з аналогами

Функціонал	Dream Drive	PDR-Online	Green Way	Easy Week	«Час»	Пояснення
Навчальні матеріали	+	+	+	-	+	Наявність матеріалів для навчання
Проходження тестів ПДР	+	+	+	-	+	Офіційні тести ПДР
Статистика навчання	+	+	+	-	+	Прогрес учнів, історія спроб, аналіз помилок

Продовження таблиці 1.3

Функціонал	Dream Drive	PDR-Online	Green Way	Easy Week	«Час»	Пояснення
Користувацькі тести	+	-	+-	-	-	Можливість створення тестів викладачами
Статистика навчання	+	+	+	-	+	Прогрес учнів, історія спроб, аналіз помилок
Управління розкладом	+	-	-	+	-	Інтерактивний розклад
Запис заняття	+	-	-	+	-	Запис учня на заняття через систему
Управління групою	+	-	+	+	-	Викладач управляє групою
CRM-функціонал	+	-	+	+	-	Інструменти для адміністрування автошколи
Оплата	+	+	+	+	+	Можливість здійснити оплату в застосунку
Сайт-візитівка	+	+	-	+	+	Чи не потрібно додатково робити лендинг
Мобільна адаптація	+	+	+	+	+	Мобільний застосунок або адаптивний вебзастосунок

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

Один із ключових етапів проєктування інформаційної системи — обґрунтований вибір архітектурних та технічних рішень, які забезпечать надійну, масштабовану та ефективну роботу вебзастосунку. У рамках реалізації системи DreamDrive було проаналізовано декілька поширених архітектурних підходів: монолітну, мікросервісну та serverless-архітектури. Монолітна архітектура, хоча й проста в реалізації, але не відповідає вимогам масштабованості та модульності, що необхідні для подальшого розвитку функціоналу. Serverless-підхід був відхилений через складність управління станом та обмежену гнучкість в інтеграції зовнішніх сервісів. У підсумку було обрано мікросервісну архітектуру, яка дозволяє розділити систему на незалежні сервіси: автентифікацію, управління користувачами, розклад занять, обробку платежів, навчальний модуль.

Для реалізації серверної частини додатку було обрано наступний технологічний стек, що включає мову програмування Python [9] та СУБД PostgreSQL [10]. Основним фреймворком для створення бекенд-логіки обрано Django [11], який у поєднанні з Django REST Framework [12] дозволяє швидко та ефективно реалізовувати RESTful API для кожного з сервісів [13]. Для маршрутизації запитів між мікросервісами та клієнтським інтерфейсом використовується окремий API Gateway, реалізований на базі FastAPI [14]. Gateway є єдиною точкою входу до системи та виконує агрегацію даних з кількох мікросервісів, використовуючи асинхронні запити, що покращує продуктивність при взаємодії з клієнтом. Всі сервіси контейнеризовані за допомогою Docker [15], що спрощує розгортання та підтримку системи у різних середовищах.

У якості системи керування базами даних застосовується PostgreSQL — стабільна, високопродуктивна реляційна база, яка підтримує складні запити, транзакції та забезпечує цілісність даних. Аутентифікація та авторизація користувачів реалізована за допомогою JWT (JSON Web Tokens) [16], що

дозволяє ефективно керувати доступом до ресурсів та зберігати контекст користувача без потреби в централізованому сховищі сесій.

Основою фронтенду є фреймворк React [17], який дозволяє створювати динамічні та інтерактивні інтерфейси користувача. Взаємодія з серверною частиною здійснюється за допомогою бібліотеки Axios [18], яка забезпечує зручний та гнучкий механізм для виконання HTTP-запитів. Для створення адаптивного та привабливого дизайну використовується Tailwind CSS [19], який дозволяє швидко стилізувати компоненти за допомогою утилітарних класів. Крім того, застосовуються сучасні бібліотека компонентів – Radix UI [20], яка надає готові до використання компоненти з можливістю гнучкого налаштування стилів . Замість класичного JavaScript [21] у проекті використовується TypeScript [22], що підвищує надійність коду, забезпечує статичну типізацію та спрощує процес рефакторингу.

В якості хмарного середовища для розгортання застосовано Docker-контейнери з подальшим деплоєм на платформі Railway [23]. Це рішення забезпечує простоту налаштування, автоматизоване масштабування, базове балансування навантаження та зручне керування версіями мікросервісів.

Обраний технічний стек є оптимальним для поточних вимог проєкту, водночас залишаючи простір для масштабування та розширення функціоналу.

1.4 Аналіз та моделювання бізнес-процесів

У цьому підрозділі буде здійснено аналіз ключових бізнес-процесів, що лежать в основі функціонування системи. З усього набору можливих сценаріїв взаємодії користувачів із системою були виділені найбільш критичні процеси, які забезпечують навчальну, адміністративну та фінансову логіку проєкту. Для їхнього опису та подальшої реалізації використано нотацію BPMN (Business Process Model and Notation), яка дозволяє чітко візуалізувати послідовність дій, ролі учасників і точки прийняття рішень [24]. Такий підхід сприяє кращому розумінню архітектури майбутнього застосунку та визначенню вузьких місць на ранніх етапах розробки. Розглянемо схеми на рисунках 1.5 – 1.7, які

представляють процеси реєстрації та оплати, запису на практичне заняття та проходження тесту.

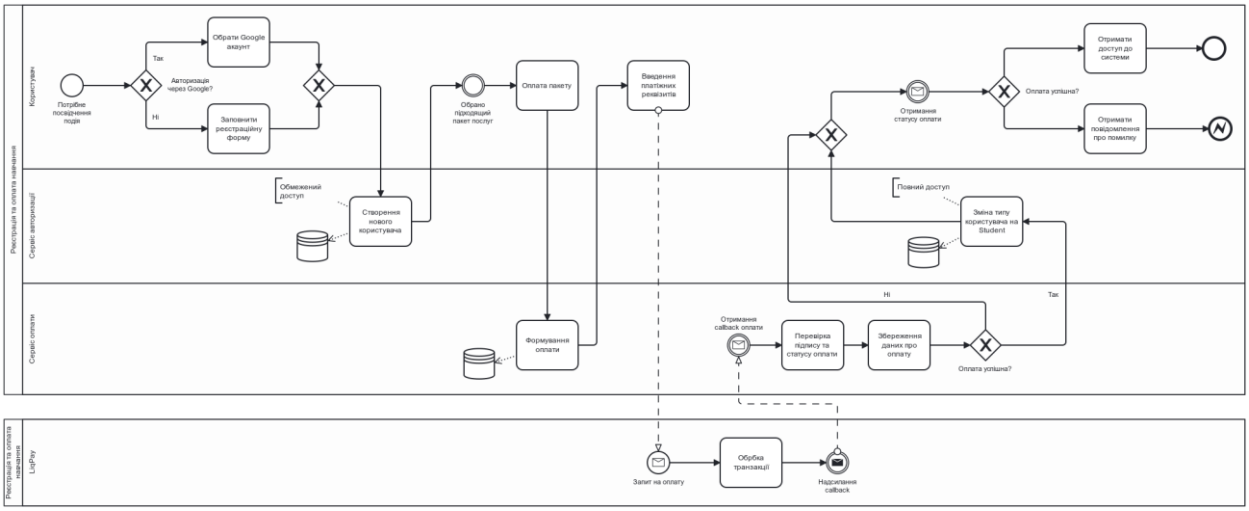


Рисунок 1.5 – Схема бізнес-процесу реєстрації та оплати навчання

Опис моделі бізнес-процесу реєстрації та оплати навчання:

– користувач заповнює форму реєстрації або реєструється через обліковий запис Google;

– сервіс авторизації додає нового користувача;

– користувач обирає бажаний тариф та натискає кнопку «Оплатити»;

– сервіс оплат ініціює створення платіжного запиту до платіжного сервісу (LiqPay [25]) та генерує форму оплати;

– користувач вводить дані у форму оплати LiqPay;

– платіжна система обробляє транзакцію та надсилає callback-запит на сервіс оплат;

– сервіс оплат перевіряє достовірність підпису та аналізує статус транзакції;

– дані про транзакцію зберігаються;

– у разі успішної оплати я та сервіс авторизації змінює тип користувача на «Student»;

– система надає користувачу доступ до функціоналу навчання;

– у разі неуспішної оплати система повідомляє користувача про невдалу транзакцію.

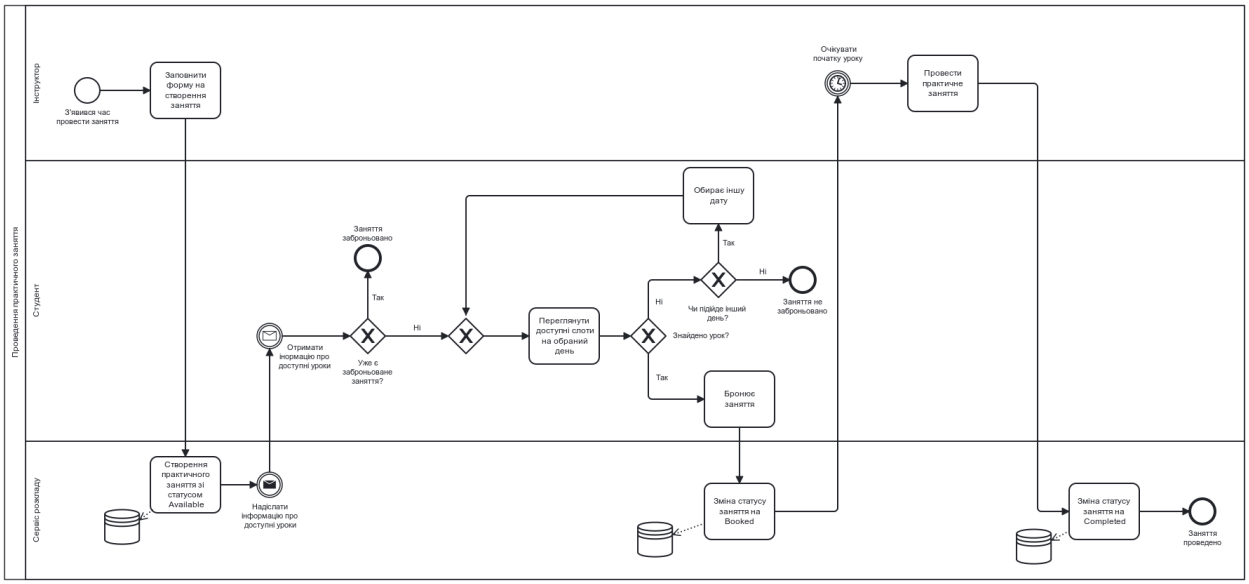


Рисунок 1.6 – Схема бізнес-процесу проведення практичного заняття

Опис моделі бізнес-процесу проведення практичного заняття:

– інструктор заповнює форму створення заняття та відправляє її;

– система створює запис уроку зі статусом «Available»;

– якщо у студента, ще немає заброньованих практичних занять, то він може переглянути доступні уроки на обраний день;

– студент обирає підходящий урок та бронює його;

– система змінює статус уроку на «Booked»;

– інструктор проводить заняття у запланований час;

– після завершення заняття система змінює статус уроку на «Completed».

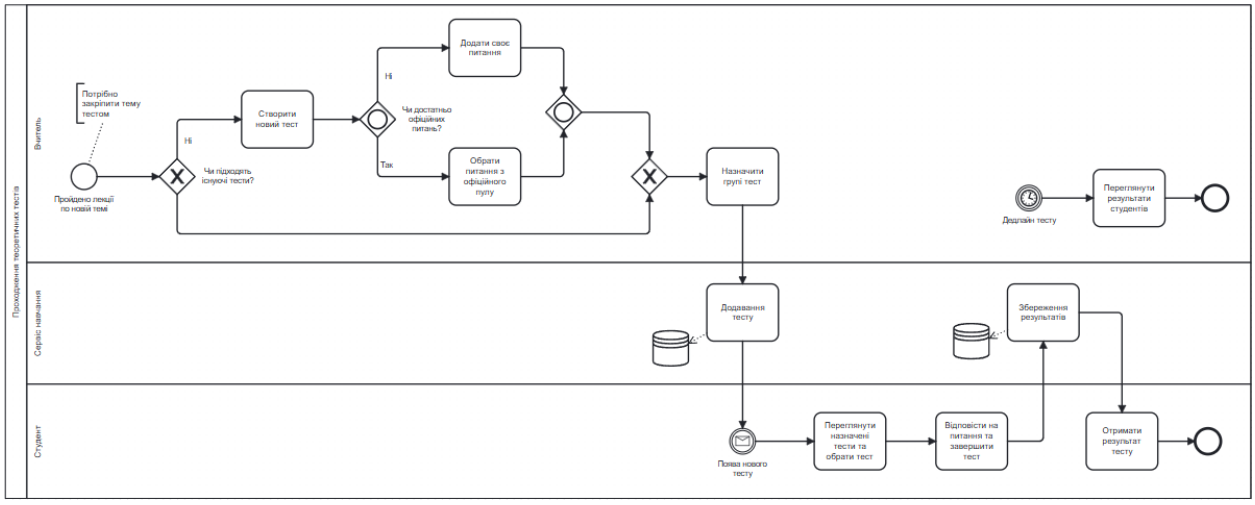


Рисунок 1.7 – Схема бізнес-процесу проходження теоретичних тестів

Опис моделі бізнес-процесу проходження теоретичних тестів:

– вчитель обирає існуючий або створює новий тест;
– якщо вчитель створює тест, то обирає офіційні питання з пулу або додає нові питання;
– система зберігає тест і робить його доступним для групи студентів;
– студенти отримують повідомлення про доступний тест;
– студент переглядає список доступних тестів та обирає тестОт;
– студент відповідає на питання тесту;
– система приймає відповіді, перевіряє їх і зберігає результати;
– студент отримує повідомлення про результати тестування;
– після дедлайну тесту вчитель переглядає результати студентів.

Висновки до розділу

У першому розділі дипломного проєкту було проведено аналіз предметної області. Було досліджено деталі організації роботи автошколи та автоматизації навчального процесу. На основі вивчення специфіки роботи закладу було сформульовано чіткі завдання для розробки вебзастосунку DreamDrive, які передбачають підвищення ефективності управління навчальним процесом.

Було виконано порівняльний аналіз існуючих програмних продуктів-аналогів, таких як PDR-Online, Green Way, EasyWeek та Автошкола Час. Було обґрунтовано потребу у платформі, яка об’єднає в собі привабливий лендинг для залучення нових клієнтів та повноцінний набір сервісів для студентів, потужні інструменти CRM для адміністрації, що працюватимуть у межах однієї системи.

Визначено, що для проєкту оптимальною є мікросервісна архітектура, яка забезпечує масштабованість, гнучкість і простоту підтримки.. Визначено стек технологій для реалізації серверної частини — фреймворки DRF та FastAPI, PostgreSQL, а також вибір клієнтського фреймворку React з підтримкою сучасних бібліотек для побудови інтерфейсу.

Крім того, було проведено моделювання ключових бізнес-процесів, що лежать в основі функціонування системи: запис на практичні заняття, проходження теоретичних тестів та здійснення оплати. Для кожного процесу розроблено покрокову послідовність дій учасників та системи, а також запропоновано візуалізацію за допомогою нотації BPMN. Це дозволяє забезпечити прозорість логіки роботи застосунку та створити передумови для подальшої автоматизації і вдосконалення.

Отже, перший розділ стане теоретичною базою для подальшої розробки, адже він чітко окреслює предметну область, визначає ключові завдання системи та технічний стек, описує основні бізнес-процеси.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Діаграма варіантів використання з ключовими акторами та основними функціями знаходиться у графічному матеріалі, креслення 1.

Таблиця 2.1 – Варіант використання UC-01

Ім’я	Перегляд головної сторінки
Ідентифікатор	UC-01
Мета	Надання користувачу загальної інформації про автошколу
Актори	Неавторизований користувач
Тригер	Користувач відкриває сайт автошколи
Передумови	-
Потік подій	На навігаційній панелі будуть доступні такі сторінки для перегляду: «Головна», «Ціни», «Філії», «Про нас», «Відгуки», «Питання» та «Контакти». Якщо виникає помилка при завантаженні сторінки або якихось компонентів (наприклад, FAQ або тарифів), відображається відповідне повідомлення. На сторінці з цінами можна обрати пакет послуг та натиснути «Обрати», користувач буде перенаправлений на сторінку авторизації для продовження оплати.
Розширення	Користувач отримує базову інформацію про автошколу та може перейти до реєстрації
Результат	Користувач ознайомився з автошколою

Таблиця 2.2 – Варіант використання UC-02

Ім’я	Реєстрація користувача
Ідентифікатор	UC-02
Мета	Надання користувачу можливості створення облікового запису
Актори	Неавторизований користувач
Тригер	Користувач бажає зареєструватися
Передумови	-

Продовження таблиці 2.2

Потік подій	Користувач натискає на кнопку входу та обирає зареєструватися, де може заповнити форму або скористатися входом через Google. В поля для реєстрації вводяться дані: пошта, прізвище та ім’я користувача, пароль та його повтор для підтвердження. Після заповнення даних користувача натискає кнопку реєстрації. Після цього з’являється повідомлення про успішну реєстрацію, і користувач перенаправляється на панель керування.
Розширення	В випадку введення не коректних даних, на екрані з’являється повідомлення з деталями помилки.
Результат	Створення сторінки користувача, перехід на сторінку входу.

Таблиця 2.3 – Варіант використання UC-03

Ім’я	Авторизація користувача
Ідентифікатор	UC-03
Мета	Надання користувачу доступу до системи з його обліковими даними
Актори	Неавторизований користувач
Тригер	Користувач бажає увійти до системи
Передумови	Користувач вже зареєстрований у системі
Потік подій	Користувач натискає на кнопку входу. Вводить електронну пошту та пароль. Натискає кнопку «Увійти». У разі успіху користувач перенаправляється до особистого кабінету. За потреби користувач може натиснути «Забув пароль» та пройти відновлення через email. За відсутності акаунту користувач може перейти на сторінку реєстрації. Після цього з’являється повідомлення про успішну авторизацію, і користувач перенаправляється на панель керування.
Розширення	Якщо дані введені некоректно, з’являється повідомлення про помилку. Якщо форма незаповнена або є помилки в полі, кнопка входу неактивна. У разі помилки авторизації через Google — повідомлення про помилку.
Результат	Користувач авторизований у системі та отримує доступ відповідно до своєї ролі

Таблиця 2.4 – Варіант використання UC-04

Ім’я	Редагування профілю
Ідентифікатор	UC-04
Мета	Надання можливості користувачу змінювати свої персональні дані в профілі
Актори	Авторизований користувач
Тригер	Користувач бажає оновити свої особисті дані або налаштування облікового запису
Передумови	Користувач входить у свій акаунт і переходить до розділу «Налаштування профілю». Вносить зміни до особистої інформації (ім'я, прізвище, контактні дані, пароль) та зберігає їх. Система перевіряє коректність даних і оновлює інформацію в базі.
Потік подій	У разі некоректного введення даних система відображає відповідні повідомлення про помилки і не зберігає зміни до усунення помилок
Розширення	Персональні дані користувача оновлюються, зміни зберігаються і відображаються в профілі користувача
Результат	Персональні дані користувача змінено

Таблиця 2.5 – Варіант використання UC-05

Ім’я	Оплата навчання
Ідентифікатор	UC-05
Мета	Надання можливості оплатити навчання для отримання повного доступу до функціоналу системи
Актори	Авторизований користувач
Тригер	Користувач натискає кнопку «Оплатити»
Передумови	Користувач повинен бути авторизований у системі
Потік подій	Користувач переходить до розділу «Платежі», обирає тарифний план та перенаправляється на платіжну платформу LiqPay. Користувач вводить реквізити картки або здійснює оплату через Google Pay чи Privat24. Користувач повертається на сайт та отримує інформацію про статус оплати та повертається на сторінку з платежами.

Продовження таблиці 2.5

Розширення	У разі помилки при оплаті система виводить повідомлення про невдачу.
Результат	Користувач отримує повний доступ до функціоналу платформи, змінюється його роль у системі на студента.

Таблиця 2.6 – Варіант використання UC-06

Ім’я	Перегляд історії оплат
Ідентифікатор	UC-06
Мета	Надання студенту можливості переглядати історію власних оплат за навчання
Актори	Студент
Тригер	Студент бажає переглянути інформацію про свої оплати
Передумови	Користувач авторизований у системі; студент оплатив навчання
Потік подій	Студент входить у свій акаунт та переходить до розділу з історією оплат. Система відображає повний перелік здійснених оплат із деталями: дата, сума, тип послуги.
Розширення	У разі відсутності оплат система показує повідомлення «Історія оплат відсутня».
Результат	Студент переглянув історію оплат

Таблиця 2.7 – Варіант використання UC-07

Ім’я	Перегляд усіх платежів
Ідентифікатор	UC-07
Мета	Надання адміністратору можливості переглядати платежі
Актори	Адміністратор
Тригер	Адміністратор бажає переглянути інформацію про оплати
Передумови	Користувач авторизований у системі як адміністатор
Потік подій	Адміністратор переходить до розділу огляду платежів. Він може подивитися статистику по платежам, а також переглянути усі здійснені платежі, включно з інформацією про дату, статус, студента, пакет послуг та ціну.
Розширення	
Результат	Адміністратор отримує доступ до повної історії платежів

Таблиця 2.8 – Варіант використання UC-08

Ім’я	Панель керування
Ідентифікатор	UC-08
Мета	Надання доступу до навчальних матеріалів з правил дорожнього руху
Актори	Авторизований користувач
Тригер	Користувач авторизується в системі
Передумови	Користувач авторизований під будь-якою роллю
Потік подій	Користувач отримує доступ панелі керування, де він може швидко отримати корисну інформацію відповідно до ролі (майбутні заняття, успішність навчання, останні відгуки). Користувач отримує набір карток, які допоможуть у навігації по основним функціональним модулям додатку: «Навчальні матеріали», «Платежі», «Тести», «Розклад», «Групи» тощо.
Розширення	Для студента, що не оплатив навчання, буде обмежений доступ та пропозиція оплатити навчання.
Результат	Користувач отримав коротке зведення інформації з різних модулів та має можливість перейти до цих модулів

Таблиця 2.9 – Варіант використання UC-09

Ім’я	Перегляд навчальних матеріалів
Ідентифікатор	UC-09
Мета	Надання доступу до навчальних матеріалів
Актори	Студент, Вчитель
Тригер	Студент або вчитель переходять до розділу з матеріалами.
Передумови	Авторизований користувач
Потік подій	Користувач отримує доступ до офіційних матеріалів ПДР розділених на три секції: правила дорожнього руху по розділам, дорожні знаки по групам та дорожня розмітка по групам.
Розширення	Перегляд матеріалів ПДР
Результат	Студент може вивчати ПДР по розділам. Вчитель ознайомився з викладеним матеріалами.

Таблиця 2.10 – Варіант використання UC-10

Ім’я	Управління навчальними матеріалами
Ідентифікатор	UC-10
Мета	Надання адміністратору можливості додавати, редагувати або видаляти навчальні матеріали
Актори	Адміністратор
Тригер	Адміністратор бажає оновити або додати нові навчальні матеріали до системи
Передумови	Користувач авторизований у системі з роллю адміністратора
Потік подій	Адміністратор входить до розділу керування навчальними матеріалами. Він переглядає наявні теми, правила, дорожні знаки та структуру матеріалів. Адміністратор може створювати нові записи, прикріплювати зображення, додавати або редагувати текст правил.
Розширення	У разі помилок при збереженні матеріалів або відсутності обов’язкових полів система показує відповідне повідомлення про помилку.
Результат	Навчальні матеріали успішно оновлені або додані, стають доступними користувачам відповідно до їхньої ролі та рівня доступу.

Таблиця 2.11 – Варіант використання UC-11

Ім’я	Перегляд своїх занять
Ідентифікатор	UC-11
Мета	Надання доступу до перегляду запланованих та пройдених занять
Актори	Студент, Вчитель
Тригер	Студент або вчитель авторизуються в системі та переходять у розділ «Мої заняття».
Передумови	Користувач авторизований у системі.
Потік подій	Студент може переглядати свої заброньовані практичні та теоретичні заняття, бачити статус кожного заняття (заплановане, завершене). Вчитель має можливість переглядати усі свої заплановані заняття.
Розширення	Якщо немає запланованих або завершених занять, відображається повідомлення «Занять немає».
Результат	Користувач переглядає список свої майбутніх занять

Таблиця 2.12 – Варіант використання UC-12

Ім’я	Інтерактивний календар
Ідентифікатор	UC-12
Мета	Надання доступу до управління своїми заняттями для викладачів та планування розкладу адміністратором
Актори	Вчитель, Адміністратор
Тригер	Адміністратор або вчитель авторизуються в системі та переходять у розділ «Мої заняття».
Передумови	Користувач авторизований у системі.
Потік подій	Вчитель може переглядати заплановані заняття у вигляді подій на календарі. Доступний перегляд поточного тижня або одного дня. Якщо натиснути на заняття, то його можна редагувати або відмінити, за умови, що до початку залишилося не менше 24 годин. При натисканні на вільний таймслот можна додати нові заняття, заповнивши форму. Адміністратор має такий же функціонал, але йому спочатку потрібно обрати вчителя, чий розклад він хоче переглянути
Розширення	При здійснені змін з’являються спливаючі повідомлення зі статусом операції
Результат	Користувач управляє своїм розкладом

Таблиця 2.13 – Варіант використання UC-13

Ім’я	Бронювання практичних занять
Ідентифікатор	UC-13
Мета	Надання студенту можливості бронювати доступні практичні заняття
Актори	Студент
Тригер	Студент бажає забронювати практичне заняття
Передумови	Користувач авторизований у системі; вчитель створив практичне заняття
Потік подій	Студент на сторінці «Мої заняття» переглядає список доступних практичних занять на обраний день. Після вибору часу студент бронює заняття.
Розширення	Якщо у студента уже є забронюване заняття, то він не зможе забронювати ще одне та отримає відповідне повідомлення
Результат	Бронювання практичного заняття

Таблиця 2.14 – Варіант використання UC-14

Ім’я	Проходження тестів
Ідентифікатор	UC-14
Мета	Надання студенту можливості проходити тести різного типу: білети як на іспиті, питання по темах та тести від викладачів
Актори	Студент
Тригер	Студент бажає пройти тестування для перевірки знань
Передумови	Користувач авторизований у системі. Викладач призначив тест
Потік подій	Студент переходить у модуль «Інтерактивні тести» та обирає тип тесту: іспит, за темами або призначений вчителем. Система завантажує відповідний набір питань і запускає тестування. Після завершення тесту студент отримує результат та може переглянути свої помилки
Розширення	Якщо під час тестування виникають технічні проблеми або недоступність питань, система відображає відповідне повідомлення з порадою повторити спробу пізніше
Результат	Студент пройшов тест

Таблиця 2.15 – Варіант використання UC-15

Ім’я	Перегляд історії тестів
Ідентифікатор	UC-15
Мета	Надання студенту можливості переглянути пройдені тести
Актори	Студент
Тригер	Студент бажає переглянути пройдені тести
Передумови	Користувач авторизований у системі. Викладач призначив тест
Потік подій	Студент переходить у модуль «Інтерактивні тести» та переходить до секції «Результати тестів», де може побачити список пройдених тестів з оцінкою, датою та часом виконання
Розширення	Щоб побачити усі результати, студент може натиснути «Переглянути усю історію»
Результат	Студент отримує зворотний зв’язок щодо рівня знань та можливість повторного проходження тесту

Таблиця 2.16 – Варіант використання UC-16

Ім’я	Перегляд результатів тесту
Ідентифікатор	UC-16
Мета	Надання студенту можливості переглянути деталі пройденого тесту
Актори	Студент
Тригер	Студент бажає переглянути пройдені тести
Передумови	Користувач авторизований у системі. Викладач призначив тест
Потік подій	Студент переходить у модуль «Інтерактивні тести» секцію «Результати тестів» та натискає один з тестів. Студент отримує розгорнуту інформацію по результатам тесту, може переглянути усі питання та свої відповіді на них. Для питань, де допущено помилку, відображається правильна відповідь та додатково додається коротке пояснення.
Розширення	Студент може повернутися на попередню сторінку натиснувши «До тестів»
Результат	Студент отримує зворотний зв’язок щодо рівня знань та можливість повторного проходження тесту

Таблиця 2.17 – Варіант використання UC-17

Ім’я	Управління тестами
Ідентифікатор	UC-17
Мета	Надання викладачу можливості створювати тести та переглядати свої тести
Актори	Вчитель
Тригер	Вчитель бажає переглянути свої тести або створити новий
Передумови	Користувач авторизований у системі та має роль викладача
Потік подій	Вчитель відкриває модуль «Інтерактивні тести». Вчитель може переглянути результати по своїм тестам та створити новий тест, натиснувши на відповідну кнопку. Він обирає питання з офіційного пулу та встановлює назву, опис, кількість питань, після чого зберігає або публікує його для студентів.
Розширення	Доступний фільтр питань по темах
Результат	Вчитель створює новий тест, доступний для проходження студентами.

Таблиця 2.18 – Варіант використання UC-18

Ім’я	Перегляд своїх груп
Ідентифікатор	UC-18
Мета	Надання викладачу можливості переглядати список закріплених навчальних груп
Актори	Вчитель
Тригер	Викладач бажає переглянути список груп, з якими він працює
Передумови	Користувач авторизований у системі та має роль викладача
Потік подій	Викладач переходить до розділу “Мої групи”. Система відображає список навчальних груп викладача. Для кожної групи вказано її назву, опис та кількість студентів,. Вчитель може перейти до перегляду деталей кожної групи, включаючи список студентів і успішність
Розширення	Якщо у викладача ще не прикріплено жодної групи, система виводить повідомлення “Немає закріплених груп”
Результат	Викладач отримує актуальну інформацію про свої навчальні групи.

Таблиця 2.19 – Варіант використання UC-19

Ім’я	Перегляд студентів
Ідентифікатор	UC-19
Мета	Надання можливості викладачу переглядати профіль студентів, закріплених за його групами
Актори	Вчитель
Тригер	Викладач бажає ознайомитися зі студентами своєї навчальної групи
Передумови	Користувач авторизований у системі та має роль викладача; обрано одну з доступних груп у розділі “Мої групи”
Потік подій	Система відображає список студентів, прикріплених до цієї групи, з основною інформацією: ПІБ, email, успішність. Вчитель може перейти до перегляду детальної інформації про кожного студента.
Розширення	Якщо у вибраній групі немає студентів, система повідомляє “У цій групі ще немає студентів”.
Результат	Викладач отримує доступ до списку студентів своєї групи з можливістю перегляду детальної інформації.

Таблиця 2.20 – Варіант використання UC-20

Ім’я	Управління користувачами
Ідентифікатор	UC-20
Мета	Надання адміністратору можливості переглядати, облікові записи користувачів системи
Актори	Адміністратор
Тригер	Адміністратор бажає переглянути користувачів системи
Передумови	Користувач авторизований у системі з роллю адміністратора
Потік подій	Адміністратор переходить до розділу керування користувачами. У таблиці він бачить перелік усіх зареєстрованих користувачів із зазначенням ролі, статусу оплати, активності та контактних даних.
Розширення	Якщо обліковий запис не знайдено або виникла помилка при оновленні даних, система відображає відповідне повідомлення
Результат	Адміністратор успішно переглядає користувачів системи

Таблиця 2.21 – Варіант використання UC-21

Use case name	Управління відгуками
Use case ID	UC-21
Goals	Надання адміністратору можливості управляти відгуками
Actors	Адміністратор
Trigger	Адміністратор бажає переглянути відгуки або змінити відображення відгуків на головній сторінці
Pre-conditions	Користувач авторизований у системі з роллю адміністратора
Flow of Events	Адміністратор переходить до розділу керування відгуками. У таблиці він бачить перелік усіх відгуків залишених користувачами, включаючи текст, оцінку, дату та автора відгука. За потреби відгук можна приховувати або публікувати на головну сторінку. Доступний пошук по тексту та автору
Extension	Якщо відгуків не знайдено, система відображає відповідне повідомлення
Post-Condition	Адміністратор успішно управляє відгуками автошколи

2.2 Розроблення функціональних вимог

Програмне забезпечення розділене на функціональні модулі. Кожен модуль має свою зону відповідальності. В таблиці 2.22 наведено загальну модель вимог, а в таблиці 2.23 наведений опис функціональних вимог до програмного забезпечення основних модулів. Матрицю трасування вимог можна побачити на рисунку 2.3.

Таблиця 2.22 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Головна сторінка	FR-1	Середній	Низький
1.1	Сторінки з цінами, філіями, зворотнім зв’язком	FR-2	Низький	Низький
1.2	Сторінки з питаннями та відгуками	FR-3	Низький	Низький
1.3	Управління відгуками	FR-4	Низький	Низький
2	Система авторизації	FR-5	Високий	Високий
2.1	Реєстрація користувача	FR-6	Високий	Високий
2.2	Авторизація користувача	FR-7	Високий	Високий
2.3	Відновлення/зміна паролю	FR-8	Середній	Середній
2.4	Вихід із системи	FR-9	Середній	Середній
2.5	Google авторизація	FR-10	Низький	Низький
3	Керування користувачами	FR-11	Високий	Високий
3.1	Редагування свого профілю	FR-12	Низький	Низький
3.2	Перегляд груп	FR-13	Високий	Середній
3.3	Перегляд студентів	FR-14	Високий	Середній
3.4	Управління користувачами	FR-15	Високий	Високий
4	Платежі	FR-16	Високий	Високий
4.1	Оплата навчання	FR-17	Високий	Високий
4.2	Перегляд історії оплат	FR-18	Середній	Низький
4.3	Перегляд усіх платежів	FR-19	Середній	Низький
5	Навчальні матеріали	FR-20	Середній	Середній
5.1	Перегляд навчальних матеріалів	FR-21	Високий	Середній
5.2	Управління матеріалами	FR-22	Високий	Низький
6	Розклад занять	FR-23	Високий	Середній
6.1	Перегляд своїх занять	FR-24	Високий	Низький
6.2	Інтерактивний календар	FR-25	Високий	Високий
6.4	Створення та редагування занять	FR-26	Високий	Середній
6.5	Бронювання практичних занять	FR-27	Високий	Середній

Продовження таблиці 2.22

№	Назва	ID Вимоги	Пріоритети	Ризики
7	Тестування	FR-28	Високий	Середній
7.1	Проходження білетів як на іспиті	FR-29	Середній	Низький
7.2	Проходження питань по темах	FR-30	Середній	Низький
7.3	Проходження тестів від вчителів	FR-31	Високий	Середній
7.4	Створення тестів	FR-32	Високий	Середній
7.5	Перегляд результатів тестів	FR-33	Високий	Низький
7.6	Статистика проходження тестів	FR-34	Середній	Середній

Таблиця 2.23 – Опис функціональних вимог

Назва	Опис
FR-1	Головна сторінка. Модуль забезпечує створення та управління головною презентаційною сторінкою автошколи. Він включає функції редагування контенту, що дозволяє оновлювати інформацію про школу, ціни, філії, відгуки, FAQ та контакти. Цей модуль створений для залучення нових клієнтів, надання їм повної інформації про послуги автошколи та покращення користувацького досвіду. Завдяки зручному інтерфейсу адміністратори можуть швидко змінювати текстову та візуальну інформацію, що робить лендинг ефективним інструментом маркетингу та комунікації.
FR-5	Система авторизації. Цей модуль відповідає за забезпечення безпечного доступу користувачів до системи. Він включає в себе функції реєстрації нових користувачів, авторизації за логіном і паролем, а також відновлення або зміну паролю у разі необхідності. Доступна авторизація за допомогою облікового запису Google. Модуль також забезпечує можливість виходу із системи для захисту сесії .

Продовження таблиці 2.23

Назва	Опис
FR-11	Керування користувачами. Цей модуль забезпечує адміністрування користувачів системи. Реалізовано управління профілями користувачів — перегляд і додавання, редагування,. Модуль підтримує управління групами користувачів, що сприяє організації навчального процесу та розподілу прав між викладачами, студентами і адміністраторами. Завдяки цьому забезпечується контроль та безпека доступу, а також ефективне управління структурою користувачів у системі.
FR-16	Платежі. Модуль відповідає за організацію фінансових операцій у системі автошколи. Він дозволяє користувачам здійснювати оплату навчання через інтегрований платіжий сервіс (LiqPay), забезпечуючи безпечність і зручність транзакцій. Крім того, модуль надає можливість перегляду історії оплат для контролю фінансових надходжень та аналізу. Цей модуль забезпечує прозорість фінансових процесів та підтримує ефективну роботу автошколи.
FR-20	Навчальні матеріали. Модуль відповідає за зберігання, організацію та відображення навчальних ресурсів, які використовуються в автошколі. Він дозволяє студентам переглядати правила дорожнього руху (ПДР) у зручному форматі, а також дає змогу додавати коментарі до окремих правил для кращого розуміння. Для викладачів і адміністраторів передбачена можливість управління навчальними матеріалами: створення, редагування та оновлення контенту.

Продовження таблиці 2.23

Назва	Опис
FR-23	Розклад занять. Модуль відповідає за організацію та управління навчальним процесом автошколи. Він дозволяє студентам переглядати актуальний розклад занять та бронювання практичних занять. Для викладачів і адміністраторів буде доступний інтерактивний календар. Він покриває функціонал перегляду, створення, редагування та зміни статусу занять.
FR-28	Тестування. Модуль призначений для оцінювання рівня підготовки студентів автошколи. Він забезпечує можливість проходження офіційних тестів, що відповідають стандартам сервісного центру, а також дозволяє створювати і проходити користувацькі тести, адаптовані під індивідуальні потреби. Вчителі можуть створювати нові тести, редагувати їх та переглядати результати своїх учнів. Модуль також надає функції аналізу успішності студентів та статистики проходження тестів, що допомагає оцінити ефективність навчання і вчасно виявити проблемні теми.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17
UC-1	+	+	+	+													
UC-2					+	+											
UC-3					+		+	+	+	+							
UC-4											+	+					
UC-5																+	+
UC-6																+	
UC-7																+	
UC-8													+			+	
UC-9																	
UC-10																	
UC-11																	
UC-12																	
UC-13																	
UC-14																	
UC-15																	
UC-16																	
UC-17																	
UC-18											+		+				
UC-19											+			+			
UC-20											+				+		
UC-21				+													

Рисунок 2.1 – Матриця трасування вимог (частина 1)

	FR-18	FR-19	FR-20	FR-21	FR-22	FR-23	FR-24	FR-25	FR-26	FR-27	FR-28	FR-29	FR-30	FR-31	FR-32	FR-33	FR-34
UC-1																	
UC-2																	
UC-3																	
UC-4																	
UC-5																	
UC-6	+																
UC-7		+															
UC-8	+	+				+	+									+	+
UC-9			+	+													
UC-10			+		+	+	+										
UC-11						+		+	+								
UC-12						+				+							
UC-13						+				+							
UC-14											+	+	+	+			
UC-15											+					+	
UC-16											+					+	
UC-17											+				+	+	+
UC-18																	
UC-19																	
UC-20																	
UC-21																	

Рисунок 2.2 – Матриця трасування вимог (частина 2)

2.3 Розроблення нефункціональних вимог

Для розробки вебзастосунку автошколи особливо важливими є такі нефункціональні вимоги, як продуктивність, доступність, безпека, зручність використання та підтримуваність. Їх реалізація є критичною для забезпечення надійної та ефективної роботи системи, комфортного користувацького досвіду й довготривалої підтримки програмного продукту.

Продуктивність є однією з ключових характеристик, оскільки система повинна швидко реагувати на запити користувачів, особливо під час проходження тестів або перегляду розкладу. Щоб задовольнити цю вимогу, застосунок буде реалізовано за принципами клієнт-серверної архітектури з використанням кешування, оптимізованих запитів до бази даних та асинхронної обробки. Основною метрикою для вимірювання продуктивності є середній час відповіді на запит, який не повинен перевищувати 1 секунди для 95% запитів у пікові години.

Доступність системи напряму впливає на її придатність до використання у будь-який момент. Оскільки учні можуть навчатися або проходити тести у зручний для себе час, вебзастосунок повинен бути доступним щонайменше 99% часу протягом місяця. Для досягнення цього буде застосовано моніторинг

роботи сервісів, автоматичне резервне копіювання даних і використання хмарної інфраструктури, яка дозволить швидко масштабувати ресурси у разі збільшення навантаження.

Безпека — ще один критичний аспект, адже система обробляє персональні дані користувачів, результати тестів та інформацію про оплату. Для її забезпечення буде реалізована багаторівнева система контролю доступу з розмежуванням ролей, автентифікація за допомогою токенів JWT та шифрування паролів із використанням bcrypt . Метриками для контролю безпеки є кількість виявлених уразливостей після автоматичного тестування, наявність сертифікатів SSL [26], а також позитивні результати перевірок безпеки, зокрема захисту від XSS [27], CSRF [28] та SQL-ін’єкцій.

Зручність використання є доволі суб’єктивною вимогою, але водночас важливою для широкого кола користувачів, зокрема студентів, інструкторів і адміністраторів. Інтерфейс повинен бути інтуїтивно зрозумілим, адаптивним до мобільних пристроїв і доступним українською мовою. Для оцінювання зручності будуть використовуватись показники часу, необхідного новому користувачу для виконання ключових дій, а також результати опитувань щодо задоволеності інтерфейсом.

Підтримуваність є основою довготривалого розвитку проєкту. Вебзастосунок буде побудований на основі мікросервісної архітектури, що дозволить легко додавати нові модулі без впливу на вже існуючі. Всі сервіси будуть покриті тестами, а код супроводжуватиметься документацією. Як метрики використовуватимуться показники покриття коду тестами, середній час внесення змін і кількість помилок, виявлених після релізу.

Отже, реалізація нефункціональних вимог забезпечить стабільну, безпечну й масштабовану роботу вебзастосунку, що відповідатиме очікуванням як адміністрації автошколи, так і кінцевих користувачів.

2.4 Аналіз системних вимог

Проведемо аналіз системних вимог, необхідних для забезпечення стабільної та ефективної роботи вебзастосунку. Вимоги охоплюють технічні характеристики як на стороні клієнта, так і на стороні сервера. Враховано типове використання системи — проходження тестів, перегляд розкладу, управління навчальними матеріалами та взаємодія з адміністративною панеллю. Застосунок має бути доступним як із сучасних персональних комп'ютерів, так і з мобільних пристроїв. Рекомендовані параметри розраховані на забезпечення комфортної роботи з кількома одночасно відкритими модулями системи. Наведені нижче таблиці 2.24 – 2.25 описують мінімальні та рекомендовані технічні вимоги до клієнтських пристроїв.

Таблиця 2.24 – Мінімальні системні вимоги

Параметр	Значення
Процесор	2-ядерний, 2.0 ГГц
Оперативна пам'ять	4 ГБ
Місце на диску (для браузера)	500 МБ вільного простору
Роздільна здатність екрану	1280x720
Браузер	Google Chrome, Firefox, Edge (останні 2 версії)
Підключення до Інтернету	10 Мбіт/с

Таблиця 2.25 – Рекомендовані системні вимоги

Параметр	Значення
Процесор	Intel Core i5 або еквівалент
Оперативна пам'ять	16 ГБ
Місце на диску (для кешу/файлів)	1 ГБ вільного простору
Роздільна здатність екрану	Full HD (1920x1080) або вища
Браузер	Google Chrome останньої версії
Підключення до Інтернету	100 Мбіт/с стабільне з'єднання

2.5 Аналіз економічних показників програмного забезпечення

Для оцінки економічних показників програмного забезпечення можна використати методику Use Case Points (UCP) разом із Technical Complexity Factor (TCF). Цей підхід дозволяє провести наближений розрахунок трудомісткості, вартості та тривалості розробки на основі складності сценаріїв використання системи та технічних характеристик.

Метод Use Case Points складається з декількох основних кроків. На першому етапі визначаються всі актори (користувачі системи або зовнішні системи, які взаємодіють із програмним забезпеченням) та класифікуються за складністю: прості, середні та складні.

Другим кроком є оцінка всіх сценаріїв використання (use cases). Кожен сценарій оцінюється за кількістю кроків або трансакцій у процесі. Залежно від складності сценарії отримують вагові коефіцієнти: простий, середній або складний. Загальна вага всіх сценаріїв дає нам Unadjusted Use Case Weight (UUCW).

Після цього розраховуються дві коригуючі величини — Technical Complexity Factor (TCF) та Environmental Complexity Factor (ECF). TCF визначається шляхом оцінки технічних факторів, таких як використання розподілених систем, продуктивність, повторне використання компонентів, безпека тощо. Кожен фактор має вагу та оцінюється за шкалою впливу, після чого розраховується загальний TCF. ECF враховує організаційні та командні фактори — досвід команди, стабільність вимог, складність програмування тощо.

Коли маємо значення UUCW, UAW (Unadjusted Actor Weight), TCF та ECF, обчислюємо загальні Use Case Points за формулою 2.1:

$$UCP = (UUCW + UAW) \times TCF \times ECF \quad (2.1)$$

Після визначення загального обсягу UCP проводиться перерахунок у трудові години, виходячи із середньої продуктивності (наприклад, 20 годин на

1 UCP). Таким чином можна приблизно оцінити обсяг робіт, термін реалізації та, знаючи вартість однієї години розробки, — загальну вартість розробки. Метод Use Case Points є ефективним у попередньому плануванні та дозволяє створити реалістичні очікування щодо ресурсів, бюджету та термінів розробки вебзастосунку.

Перейдемо до обчислення.

Таблиця 2.26– Оцінка складності акторів

Актор	Тип взаємодії	Категорія	Вага	К-сть	Разом
Неавторизований користувач	HTTP-запити (Головна сторінка)	Середній	2	1	2
Авторизований студент	Веб-інтерфейс з логікою (оплати, розклад, тести)	Складний	3	1	3
Інструктор	Веб-інтерфейс з доступом до учнів, розкладу	Складний	3	1	3
Адміністратор	Веб-інтерфейс з повним контролем системи	Складний	3	1	3

Всього: 11 (UAW)

Таблиця 2.27 – Оцінка складності Use Case

Тип сценарію	Кількість кроків	Вага	Кількість	Разом
Прості	≤3 кроки	5	11	55
Середні	4–7 кроків	10	7	70
Складні	>7 кроків	15	4	60

Всього: 185 (UUCW)

Таблиця 2.28 – Оцінка технічних факторів

№	Фактор	Вага	Оцінка	Внесок
T1	Розподілена система	2	2	4
T2	Продуктивність	1	3	3
T3	Кінцева зручність	1	5	5
T4	Комплексна логіка	1	4	4
T5	Повторне використання коду	1	3	3
T6	Простота інсталяції	0.5	2	1
T7	Портованість	2	1	2
T8	Легка зміна	1	3	3
T9	Паралельне використання	1	2	2
T10	Безпека	1	4	4
T11	Доступність	1	4	4
T12	Підтримка транзакцій	1	3	3
T13	Особлива логіка	1	2	2

Всього: 40 (TF)

Обчислимо Technical Complexity Factor (формула 2.2):

$$TCF = 0.6 + (0.01 \times \sum TF) = 0.6 + 0.01 \times 40 = 1.0 \quad (2.2)$$

Таблиця 2.29 – Оцінка факторів середовища

№	Фактор	Вага	Оцінка	Внесок
E1	Досвід з UML	1.5	4	6
E2	Досвід у застосунку	0.5	3	1.5
E3	Досвід у розробці	1	4	4
E4	Мотивація	1	5	5
E5	Аналітичні здібності	0.5	5	2.5

Продовження таблиці 2.29

№	Фактор	Вага	Оцінка	Внесок
E6	Командна підтримка	1	2	2
E7	Стабільність вимог	2	4	8
E8	Частковий доступ до користувача	-1	5	-5

Всього: 24.5 (EF)

Тоді за формулою 2.3 знайдемо ECF:

$$ECF = 1.4 + (-0.03 \times \sum EF) = 1.4 - 0.03 \times 24.5 = 0.665 \quad (2.3)$$

Отже, ми отримали:

– UAW (Unadjusted Actor Weight) = 11;
– UUCW (Unadjusted Use Case Weight) = 185;
– TCF (Technical Complexity Factor) = 1.0;
– ECF (Environmental Factor) = 0.665.

Розрахунок Unadjusted Use Case Points (UUCP) [формула 2.4]:

$$UUCP = UAW + UUCW = 11 + 185 = 196 \quad (2.4)$$

Розрахунок Use Case Points (UCP) [формула 2.5]:

$$UCP = UUCP \times TCF \times ECF = 236 \times 1.0 \times 0.665 = 130.34 \quad (2.5)$$

Тепер можемо перейти до обчислення наступних метрик:

– трудомісткість;
– вартість розробки;
– тривалість розробки.

Для оцінки трудомісткості проєкту зазвичай беруть середнє значення трудомісткості на 1 UCP (продуктивність), було обрано 20 людино-годин на 1 UCP (формула 2.6):

$$\text{Трудомісткість} = UCP \times 20 = 130.34 \times 20 = 2606.8 \text{ людино} - \text{годин} \quad (2.6)$$

Щоб оцінити вартість, потрібно знати середню вартість людино-години розробника. Беремо ставку — 6 доларів за годину, що приблизно дорівнюватиме 1000 доларам на місяць (формула 2.7).

$$\text{Вартість} = \text{Трудомісткість} \times \text{Ставка} = 2606.8 \times 6 = 15640 \text{ доларів} \quad (2.7)$$

Тривалість оцінюється залежно від кількості розробників у команді та робочих годин на тиждень. У нас один розробник, який працює 40 годин на тиждень (формули 2.8-2.9).

$$\text{Тривалість} = \frac{\text{Трудомісткість}}{\text{Кількість розробників} \times \text{Години на тиждень}} \quad (2.8)$$

$$\text{Тривалість} = \frac{2606.8}{1 \times 40} = 65{,}17 \text{ тижнів} \quad (2.9)$$

Метод Use Case Points (UCP) дозволив провести обґрунтовану та кількісну оцінку трудомісткості, вартості та тривалості розробки вебзастосунку на основі складності сценаріїв використання, типів акторів та технічних і екологічних факторів. Завдяки цьому підходу, не маючи детального плану чи прототипу, ми на етапі аналізу вимог отримали реалістичне уявлення про масштаби проєкту, що сприяє кращому плануванню ресурсів та визначенню доцільних строків реалізації. Метод виявився ефективним для попереднього прогнозування зусиль, необхідних для створення повноцінної системи.

2.6 Постановка завдання на розробку програмного забезпечення

У рамках дипломного проєкту розробляється вебзастосунок DreamDrive для автоматизації ключових процесів в автошколі. Система має забезпечити ефективне управління навчальним процесом, комунікацією між студентами, викладачами та адміністраторами, а також доступ до навчальних матеріалів та

тестування знань. Застосунок має підтримувати багаторольовий доступ з урахуванням специфіки функціоналу для кожної категорії користувачів.

Метою проєкту є підвищення ефективності управління освітніми та адміністративними процесами автошколи шляхом консолідації розрізнених цифрових інструментів в єдину інтегровану систему. Очікуваний результат — зменшення кількості використовуваних програмних засобів із трьох до одного, що дозволить скоротити час адміністрування, покращити взаємодію між користувачами та підвищити прозорість освітнього процесу.

Застосунок має підтримувати авторизацію з розподілом прав доступу відповідно до ролей користувачів (студент, викладач, адміністратор). Студенти повинні мати можливість переглядати свій розклад занять, проходити тести та отримувати доступ до навчальних матеріалів. Викладачі можуть створювати та редагувати заняття в інтерактивному календарі, переглядати результати тестування учнів і контролювати процес навчання. Адміністратор керує всіма користувачами, здійснює налаштування матеріалів, складає розклад, а також має доступ до фінансової інформації, зокрема історії оплат. У системі реалізується модуль для вивчення правил дорожнього руху за темами та тестування у форматі офіційних білетів. Крім того, вебзастосунок повинен мати зручний інтерфейс, адаптований до мобільних пристроїв, і відповідати базовим вимогам безпеки, зокрема захисту персональних даних.

Висновки до розділу

У даному розділі було проаналізовано різні види вимог для створення вебзастосунку DreamDrive. Розпочато з розгляду типових сценаріїв використання системи (Use Cases) ключовими ролями — студентами, викладачами та адміністраторами. На основі цих сценаріїв були сформульовані функціональні вимоги, що охоплюють реєстрацію, управління заняттями, тестування знань, доступ до навчальних матеріалів, а також адміністрування системи та облік платежів. Далі були визначені нефункціональні вимоги до надійності, безпеки, зручності інтерфейсу,

масштабованості та підтримуваності застосунку. У підрозділі аналізу системних вимог розглянуто технічні обмеження, наведено мінімальні та рекомендовані параметри для користувацьких пристроїв. Економічна оцінка виконано методом Use Case Points із урахуванням технічних і екологічних факторів, що дозволило розрахувати орієнтовну трудомісткість, вартість та тривалість розробки проєкту. Завершальною частиною стало формулювання загальної постановки задачі, яка окреслює цілі створення застосунку та ключові завдання, що мають бути вирішені в результаті його реалізації.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

У проєкті DreamDrive застосовано мікросервісну архітектуру, яка передбачає поділ системи на набір незалежних сервісів із чітко визначеними зонами відповідальності. Такий підхід дозволяє досягти високої гнучкості розробки, спрощує масштабування окремих компонентів, а також покращує підтримку та розвиток системи в довгостроковій перспективі.

Архітектура системи складається з п’яти мікросервісів: сервіс авторизації і користувачів, сервіс оплат, сервіс розкладу, сервіс навчання, та сервіс управління головною сторінкою. Перша складова відповідає за автентифікацію користувачів, авторизацію та управління обліковими записами. Забезпечує реалізацію ролей користувачів (студент, викладач, адміністратор) і контролює доступ до відповідних функцій та ресурсів системи. Цей сервіс є базовим для безпеки системи та гарантує коректну ідентифікацію користувачів.

Сервіс оплат відповідає за інтеграцію з платіжною системою, забезпечує функціонал оплати курсів, збереження і обробку історії транзакцій. Цей сервіс гарантує безпечне та зручне проведення фінансових операцій.

Керування розкладом занять бере на себе відповідний мікросервіс. Він забезпечує функціонал для студентів з можливістю запису на заняття, а для викладачів та адміністраторів — планування теоретичних та практичних занять. Цей сервіс автоматизує процес організації навчального часу та підтримує актуальність розкладу.

Навчальний модуль призначений для управління матеріалами. Організовує структуру навчальних тем, правил і дорожніх знаків, а також забезпечує проходження офіційних тестів і білетів. Дозволяє інтерактивно працювати з навчальним контентом та перевіряти знання.

Доступ до загальнодоступної інформації без необхідності автентифікації користувача забезпечує останній мікросеріс. Серед іншого, надає дані про курси, відповіді на поширені запитання (FAQ), контактні дані та іншу інформацію, що підтримує користувачів і потенційних клієнтів. Для адміністратора реалізує функціонал часткового управління контентом сторінок.

Взаємодія між клієнтським додатком (frontend) та мікросервісами здійснюється через API Gateway — центральний компонент, який маршрутизує запити, реалізує контроль доступу та агрегує сервіси, спрощуючи комунікацію. Важливо відзначити, що мікросервіси у системі функціонують незалежно один від одного і не мають прямих зв'язків між собою, що підвищує їх автономність і зменшує зв'язність системи.

Для більш наочного опису архітектури використаємо діаграму C4, яка дозволяє описати систему на різних рівнях абстракції. Назва «C4» походить від чотирьох рівнів діаграм, які допомагають чітко і послідовно уявити структуру програмного продукту. Почнемо з Context Diagram (Діаграма контексту) [рисунок 3.1].

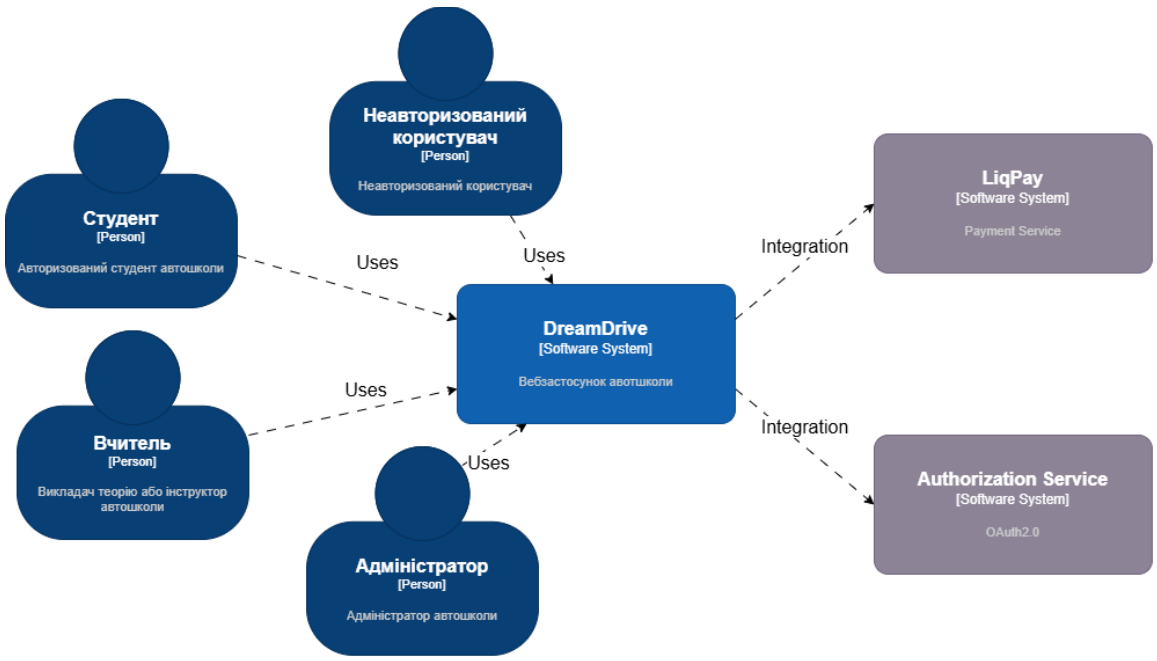


Рисунок 3.1 – C4. Діаграма контексту (1 рівень)

На найвищому рівні можемо побачити чотири типи користувачів та інтеграції з двома зовнішніми системами. Вебзастосунок зображено одним блоком, щоб дізнатися, що всередині – потрібно перейти до наступного рівня. Container Diagram (Діаграма контейнерів) показує основні «контейнери» всередині системи, рисунок 3.2.

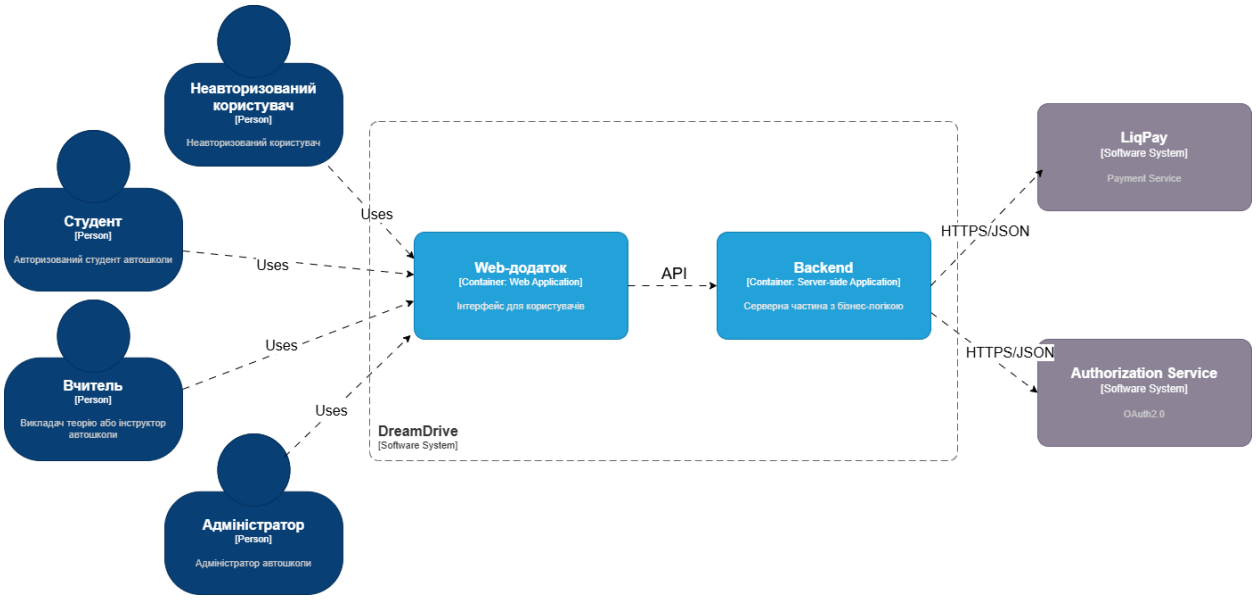


Рисунок 3.2 – C4. Діаграма контейнерів (2 рівень)

Отже, на рівні контейнерів маємо клієнт-серверну архітектуру. Користувачі взаємодіють з інтерфейсом веб-додатку, який у свою чергу передає їх запити до серверу через API. Бекенд спілкується з зовнішніми системами за допомогою протоколу HTTPS та отримує від них дані у форматі JSON. Наш контейнер Backend досі вигядає доволі узагальнено, тому пропоную розбити його на компоненти. Для цього перейдемо до діаграми компонентів, що деталізує окремі контейнери на компоненти — підсистеми, модулі, сервіси всередині кожного контейнера.

На рисунку 3.3 бачимо мікросервісну архітектуру: 5 незалежних сервісів, у кожного своя база даних. Клієнтський застосунок має єдину точку входу на сервері – API Gateway, він допомагає знайти маршрут до потрібного мікросервісу або навіть зробити агрегацію даних у результаті кількох REST API запитів до мікросервісів.

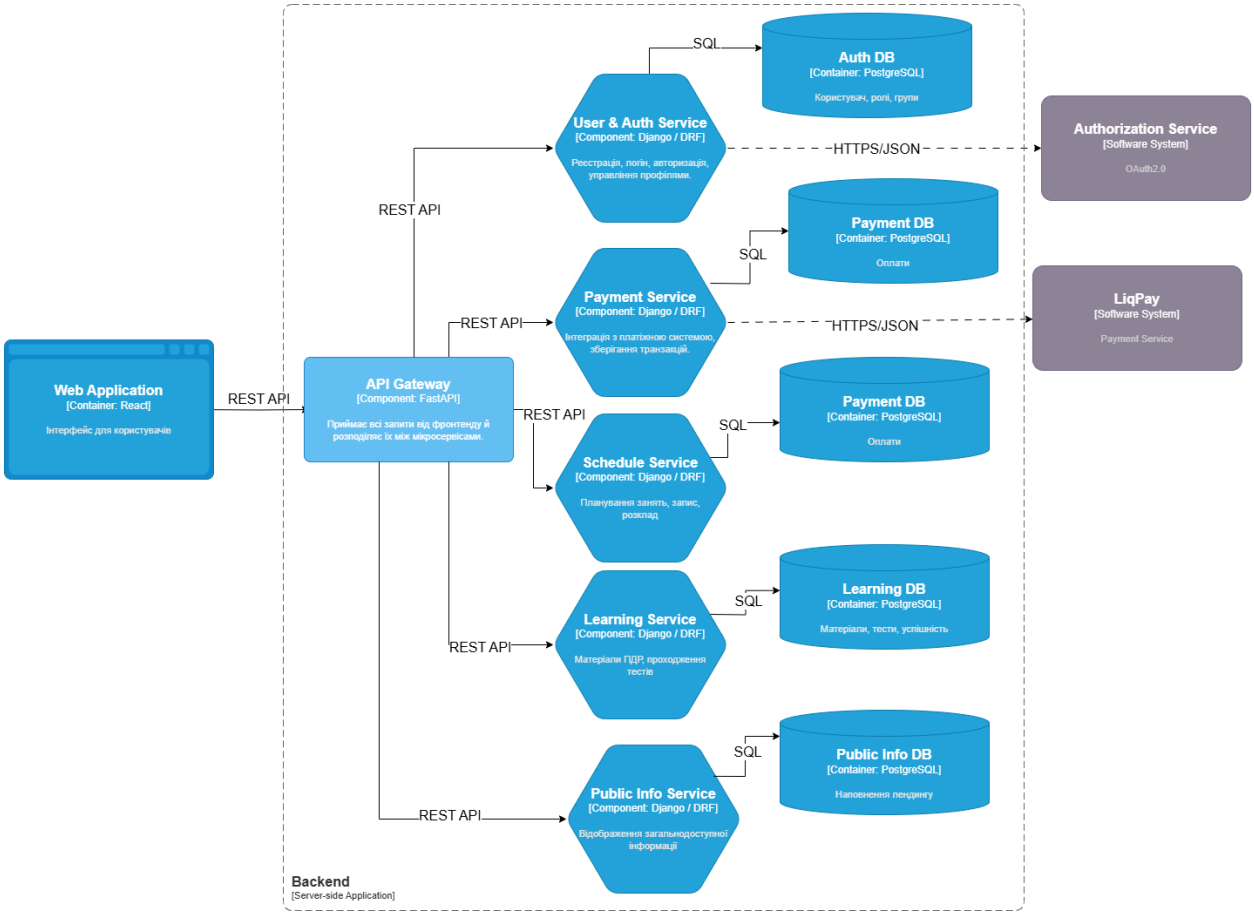


Рисунок 3.3 – С4. Діаграма компонентів (3 рівень)

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

У процесі розробки вебзастосунку було прийнято низку архітектурних рішень, які забезпечують функціонування системи. Одним із ключових аспектів стала організація управління користувачами та автентифікації. Система зберігає облікові записи користувачів локально та реалізує самостійну авторизацію без використання механізмів single sign-on (SSO). Такий підхід є доцільним у контексті роботи автошколи, де реєстрація нових користувачів здійснюється через адміністратора або самостійно студентами. Кожен користувач має свою роль (студент, викладач, адміністратор), що визначає рівень доступу до функціоналу. Процес входу в систему реалізовано за допомогою JWT (JSON Web Token), що дозволяє ефективно управляти сесіями без необхідності зберігати стан на сервері. Це спрощує масштабування

сервісу та забезпечує безпечну передачу авторизаційної інформації між клієнтом і сервером.

У межах архітектури передбачено інтеграцію із зовнішніми системами, що розширює функціональність платформи та забезпечує зручність користування для кінцевих користувачів. Такі інтеграції реалізуються за допомогою сучасних протоколів взаємодії.

Для зручної авторизації в майбутньому передбачається підтримка інтеграції з зовнішніми постачальниками автентифікації через протокол OAuth 2.0 [29]. Це дозволить студентам та викладачам входити до системи за допомогою сторонніх акаунтів (наприклад, Google або Facebook), підвищуючи зручність і швидкість реєстрації. Ще одним важливим компонентом є інтеграція з платіжною системою LiqPay, яка забезпечує можливість здійснення онлайн-оплат за навчальні послуги. Передача платіжних даних відбувається через захищені HTTP(S) API з використанням формату JSON, що гарантує безпечну та стандартизовану комунікацію між сервісами.

Для забезпечення належної якості вебзастосунку DreamDrive було визначено низку нефункціональних вимог, серед яких ключовими є продуктивність, доступність, безпека, зручність використання та підтримуваність. Їх виконання гарантує стабільну роботу системи, позитивний досвід взаємодії для користувачів та можливість подальшого розвитку застосунку.

Продуктивність досягається завдяки мікросервісній архітектурі, що дозволяє ефективно розподіляти навантаження між сервісами. Для зменшення часу відповіді застосовується кешування на рівні API Gateway і баз даних, а також використання асинхронних механізмів обробки запитів. Доступність підтримується шляхом автоматизованого моніторингу роботи сервісів. Безпека забезпечується використанням JWT для авторизації, шифруванням паролів через bcrypt, а також обмеженням доступу до ресурсів відповідно до ролі користувача. Важливим є й захист від типових атак, таких як SQL-ін’єкції, XSS і CSRF. Зручність використання реалізується через інтуїтивно зрозумілий

інтерфейс, адаптований до мобільних пристроїв і орієнтований на україномовну аудиторію. Для підтримуваності в архітектурі застосовано принцип розділення відповідальностей між сервісами, що спрощує тестування, оновлення і додавання нового функціоналу. Код супроводжується документацією, а основні компоненти покриваються модульними тестами, що забезпечує стабільність при змінах.

Для зберігання структурованих даних обрано реляційний тип бази даних. Це зумовлено необхідністю забезпечення чітких зв’язків між сутностями (користувачі, розклад, заняття, оплати, правила ПДР, тести тощо), підтримки цілісності даних та виконання складних запитів із використанням JOIN-операцій. Реляційна модель ідеально підходить для більшості бізнес-процесів системи, де важливо контролювати зв’язки між таблицями та забезпечити транзакційність.

У перспективі, з огляду на наявність великої кількості медіа-файлів (зображення дорожніх знаків, ілюстрації до питань тощо), планується додати окреме сховище для об'єктів (object storage), наприклад, Amazon S3 [30]. Таке сховище краще підходить для зберігання великих бінарних файлів, забезпечує масштабованість і зручний доступ через HTTP(S), не навантажуючи основну базу даних. Зображення при цьому зберігатимуться поза базою, а в самій БД будуть лише посилання на відповідні ресурси.

Для реалізації серверної частини вебзастосунку обрано мову програмування Python у поєднанні з фреймворком Django, який надає зручні засоби для організації мікросервісної архітектури, роботи з ORM, автентифікації та створення REST API через Django REST Framework. Django дозволяє швидко створювати надійні сервіси з чітко визначеною відповідальністю, що особливо важливо при побудові масштабованої системи. Взаємодія між клієнтською частиною та мікросервісами організована через API Gateway, реалізований на базі FastAPI, що забезпечує швидку маршрутизацію HTTP-запитів, обробку авторизації та централізацію доступу до сервісів.

Клієнтська частина реалізована з використанням React і TypeScript, що дозволяє створювати динамічний, адаптивний та масштабований інтерфейс з урахуванням потреб різних типів користувачів — студентів, викладачів і адміністраторів. Для зберігання даних застосовується реляційна база даних PostgreSQL, яка добре підходить для роботи з транзакційною інформацією, зокрема розкладами занять, тестами, результатами й платіжними даними.

3.3 Конструювання програмного забезпечення

3.3.1 Опис структури бази даних

В якості системи управління базами даних використовується PostgreSQL. База даних серверу призначена для зберігання інформації про користувачів, їх профілі, навчальний процес, розклад занять, платежі та публічний контент системи. Опис таблиць бази даних наведено у таблицях 3.1-3.22. Модель бази даних наведена у графічному матеріалі, креслення 2.

Таблиця 3.1 – Опис таблиці Review (Public Info Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер відгуку
filial_id	INT	Посилання на запис у таблиці Filial
student	INT	Посилання на запис у таблиці StudentProfile
student_name	VARCHAR(100)	Прізвище та ім'я студента
rating	INT	Оцінка у відгуку
text	TEXT	Текст відгуку
created_at	DATETIME	Дата та час створення
is_active	BOOLEAN	Прапорець, що визначає, чи видимий відгук

Таблиця 3.2 — Опис таблиці Filial (Public Info Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер філії
name	VARCHAR(100)	Назва філії
address	VARCHAR(100)	Адреса філії
city	VARCHAR(100)	Місто філії
phone	VARCHAR(20)	Контактний номер телефону філії
email	VARCHAR(100)	Контактна електронна адреса
latitude	DECIMAL(8,6)	Географічна широта філії
longitude	DECIMAL(9,6)	Географічна довгота філії

Таблиця 3.3 — Опис таблиці FAQ (Public Info Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер питання
question	TEXT	Текст питання
answer	TEXT	Текст відповіді
sort_order	INT	Порядок сортування для відображення
is_active	BOOLEAN	Прапорець, що визначає, чи активне це питання

Таблиця 3.4 — Опис таблиці LandingElement (Public Info Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер елемента
section	VARCHAR(100)	Назва секкції лендингу
name	VARCHAR(100)	Назва елемента
content	TEXT	Контент або опис елемента
sort_order	INT	Порядок сортування для відображення
image_url	VARCHAR(255)	URL-адреса зображення
is_active	BOOLEAN	Прапорець, що визначає, чи активний елемент

Таблиця 3.5 — Опис таблиці PricePlan (Public Info Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер тарифного плану
title	VARCHAR(100)	Назва тарифного плану
description	TEXT	Опис тарифного плану
price	DECIMAL(6,2)	Вартість тарифного плану
currency	VARCHAR(10)	Валюта ціни (наприклад, "UAH", "USD")
is_active	BOOLEAN	Прапорець, що визначає, чи активний тарифний план

Таблиця 3.6 — Опис таблиці User (Auth / User Service)

Назва поля	Тип даних	Опис
id	INT	Унікальний ідентифікатор користувача
first_name	VARCHAR(100)	Ім’я користувача
last_name	VARCHAR(100)	Прізвище користувача
email	VARCHAR(100)	Електронна пошта користувача
date_of_birth	DATE	Дата народження
phone	VARCHAR(20)	Номер телефону користувача
password	VARCHAR(255)	Хеш пароля
address	VARCHAR(255)	Адреса проживання користувача
about_me	TEXT	Додаткова інформація, яку користувач хоче додати
role	VARCHAR(50)	Роль користувача (наприклад, admin, student, teacher)
is_active	BOOLEAN	Прапорець, що визначає, чи активний користувач
is_stuff	BOOLEN	True для адміністратора
is_paid	BOOLEAN	Чи оплатив студент навчання. Для вчителів та адмінів false.
created_at	DATETIME	Дата й час створення облікового запису
updated_at	DATETIME	Дата й час останнього оновлення запису

Таблиця 3.7 — Опис таблиці Group (Auth / User Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер групи
name	VARCHAR(100)	Назва групи
description	TEXT	Опис групи (зокрема, графік роботи)
driving_category	VARCHAR(12)	Категорія водійського посвідчення (наприклад, B, C)
teacher_id	INT	Посилання на запис у таблиці TeacherProfile
type	VARCHAR(20)	Група теорії або практики
filial_id	INT	Філіал, до якого відноситься група.

Таблиця 3.8 — Опис таблиці StudentProfile (Auth / User Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер студентського профілю
user_id	INT	Посилання на запис у таблиці User
group_id	INT	Посилання на запис у таблиці Group

Таблиця 3.9 — Опис таблиці TeacherProfile (Auth / User Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер викладацького профілю
user_id	INT	Посилання на запис у таблиці User
type	VARCHAR(50)	Тип викладача (теорія або практика)

Таблиця 3.10 — Опис таблиці Payment (Payment Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер платежу
student_id	INT	Посилання на запис у таблиці StudentProfile
amount	DECIMAL(8,2)	Сума платежу
currency	VARCHAR(10)	Валюта платежу (наприклад, "UAH", "USD")
status	VARCHAR(50)	Статус платежу (наприклад, "pending", "completed")
payment_method	VARCHAR(50)	Метод оплати (наприклад, "card", "cash", "bank")
description	VARCHAR(255)	Коментар до платежу
created_at	DATETIME	Дата створення платежу
confirmed_at	DATETIME	Дата підтвердження платежу

Таблиця 3.11 — Опис таблиці TheoryLesson (Schedule Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер
group_id	INT	Посилання на запис у таблиці Group
teacher_id	INT	Посилання на TeacherProfile
filial_id	INT	Посилання на запис у таблиці Filial
title	VARCHAR(100)	Тема заняття
start_time	TIME	Час початку заняття
duration	INTERVAL	Тривалість заняття
is_online	BOOLEAN	Якщо 1, то це онлайн заняття
created_at	DATETIME	Дата створення запису
status	VARCHAR(10)	Статус: scheduled, cancelled, completed

Таблиця 3.12 — Опис таблиці PracticeLesson (Schedule Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер практичного заняття
title	VARCHAR(100)	Тема заняття
status	VARCHAR(10)	Статус заняття: available, booked, completed, cancelled.
student_id	INT	Посилання на запис у таблиці StudentProfile. Якщо заняття вільне, то null.

Продовження таблиці 3.12

Назва поля	Тип даних	Опис
teacher_id	INT	Посилання на запис у таблиці TeacherProfile
filial_id	INT	Посилання на запис у таблиці Filial
start_time	TIME	Час початку заняття
duration	INTERVAL	Тривалість заняття
car	VARCHAR(50)	Ідентифікатор або марка авто, що використовується
location	VARCHAR(255)	Адреса локації, на якій буде проведено заняття
created_at	DATETIME	Дата створення запису

Таблиця 3.13 — Опис таблиці RoadVisualElement (Learning Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер елемента
element_id	VARCHAR(10)	Номер у чинних ПДР України
title	VARCHAR(255)	Назва елемента
text	TEXT	Опис елемента
image_url	VARCHAR(255)	URL зображення елемента
group_id	INT	Посилання на запис у таблиці RoadVisualGroup

Таблиця 3.14 — Опис таблиці RoadVisualGroup (Learning Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер групи
number	INT	Номер групи у ПДР України
title	VARCHAR(255)	Назва групи візуальних елементів
type	VARCHAR(10)	Тип елементу: sign, marking.

Таблиця 3.15 — Опис таблиці RuleSection (Learning Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер секції правил
title	VARCHAR(255)	Назва секції
number	INT	Номер секції у ПДР України

Таблиця 3.16 — Опис таблиці Rule (Learning Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер правила
section_id	INT	Посилання на запис у таблиці RuleSection
rule_number	VARCHAR(255)	Назва правила
rule_text	TEXT	Текст правила

Таблиця 3.17 — Опис таблиці Question (Learning Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер питання
section_id	INT	Посилання на запис у таблиці RuleSection
ticket_number	INT	Номер офіцйного білету
question_number	INT	Номер питання
text	TEXT	Текст питання
image_url	VARCHAR(255)	URL зображення (якщо є)
reply_text	TEXT	Пояснення до правильної відповіді
rule_id	VARCHAR(255)	Посилання на відповідне правило (текстове) (якщо є)

Таблиця 3.18 — Опис таблиці Answer (Learning Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер відповіді
question_id	INT	Посилання на запис у таблиці Question
text	VARCHAR(1024)	Текст відповіді
is_correct	BOOLEAN	Прапорець, що визначає, чи відповідь правильна

Таблиця 3.19 — Опис таблиці Ticket (Learning Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер відповіді
number	INT	Номер білету
driving_category	VARCHAR(5)	Білет призначений для якої категорії транспортних засобів (A, B, BE тощо)

Таблиця 3.20 — Опис таблиці TicketQuestion (Learning Service)

Назва поля	Тип даних	Опис
ticket_id	INT	Посилання на Ticket
question_id	INT	Посилання на Question

Таблиця 3.21 — Опис таблиці TicketTestSession (Learning Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер тестової сесії
student_id	INT	Посилання на запис у таблиці StudentProfile
ticket_id		Посилання на запис у таблиці Ticket
started_at	DATETIME	Дата та час початку сесії
completed_at	DATETIME	Дата та час завершення сесії
mistake_count	INT	Кількість неправильних відповідей
is_passed	BOOLEAN	Чи здано екзамено? (1 – здано, 0 – не здано)

Таблиця 3.22 — Опис таблиці TicketTestAnswer (Learning Service)

Назва поля	Тип даних	Опис
id	INT	Ідентифікаційний номер тестової сесії
session_id	INT	Посилання на запис у таблиці TicketTestSession
selected_answer_id	INT	Посилання на запис у таблиці Answer
answer_at	DATETIME	Дата та час відповіді

3.3.2 Опис утиліт, бібліотек та іншого стороннього програмного забезпечення

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.23. Серед них можна виділити середовища для розробки, тестування та розгортання програмного забезпечення.

Таблиця 3.23 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	PyCharm	Основне середовище розробки серверної частини застосунку, зокрема мікросервісів на Django.
2	Visual Studio Code	Використовується для розробки клієнтської частини застосунку на React + TypeScript.

№ п/п	Назва утиліти	Опис застосування
3	Postman	Програмне забезпечення для тестування REST API-запитів між клієнтом та мікросервісами.
4	Docker Desktop	Використовується для контейнеризації мікросервісів, спрощення розгортання та ізоляції середовищ.
5	Google Chrome	Основний веб-браузер, що застосовується для перегляду, відлагодження та тестування інтерфейсу користувача

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

3.4 Аналіз безпеки даних

Особлива увага приділяється забезпеченню безпеки персональних даних користувачів, а також захисту від основних кіберзагроз. Оскільки система працює з конфіденційною інформацією — акаунтами студентів, викладачів та адміністрації, а також платіжними даними, — впроваджено комплекс заходів, спрямованих на зменшення ризиків несанкціонованого доступу, втрати чи підміни даних.

Управління вразливостями реалізується через постійне оновлення бібліотек та фреймворків (зокрема Django, FastAPI), а також регулярну перевірку залежностей за допомогою таких інструментів, як pip-audit або safety. Завдяки модульній структурі застосунку, кожен сервіс може оновлюватися незалежно, що дозволяє оперативно закривати виявлені вразливості.

Захист API забезпечується за допомогою JWT-токенів, а комунікація між клієнтом та сервером — через протокол HTTPS. У Django використовуються вбудовані механізми захисту від CSRF-атак, а також здійснюється валідація вхідних даних з метою запобігання SQL-ін’єкціям та XSS-атакам.

Контейнери, в яких розгортаються мікросервіси, побудовані на основі офіційних мінімалістичних Docker-образів. Запуск додатків здійснюється без root-доступу, що дозволяє зменшити ризики у разі компрометації одного з сервісів. Конфіденційні дані, такі як секретні ключі та паролі до баз даних, зберігаються у .env-файлах, що не включаються до репозиторію.

Додатково реалізовано хешування паролів за допомогою надійного алгоритму (PBKDF2, стандарт Django), обмеження CORS-політик, а також передбачено можливість розширення безпекових заходів у разі масштабування системи або переходу до хмарної інфраструктури. Таким чином, DreamDrive забезпечує базовий рівень безпеки відповідно до сучасних вимог, з урахуванням подальшої можливості вдосконалення.

Висновки до розділу

Отже, було розглянуто процес проєктування та реалізації вебзастосунку DreamDrive. Основну увагу було приділено архітектурі системи та опису її компонентів. Мікросервісна клієнт-серверна архітектура була представлена за допомогою C4-моделі, що надало чітке уявлення про компоненти системи, їх взаємодію та внутрішню організацію.

Важливою частиною стало обґрунтування вибору мов програмування, фреймворків і бази даних. Такі інструменти, як Python, Django, FastAPI та PostgreSQL, були обрані з огляду на їхню популярність, гнучкість та відповідність вимогам до створення сучасних вебсервісів..

Окремо було розроблено логіку роботи основних сервісів і побудовано структуру бази даних, яка забезпечує зберігання інформації про користувачів,

навчальні матеріали, розклад занять, результати тестування та інші важливі об'єкти предметної області. Структура бази даних орієнтована на забезпечення цілісності, зв’язності та зручного доступу до даних.

Також було приділено увагу питанням безпеки: реалізовано багаторівневу систему автентифікації та авторизації, обмеження доступу на основі ролей, захист API, шифрування паролів, а також враховано особливості безпечного використання контейнеризації та зберігання конфіденційних даних у хмарному середовищі.

Отже, у цьому розділі було закладено надійну технічну основу вебзастосунку DreamDrive, що поєднує сучасні підходи до архітектури, реалізації бізнес-логіки та забезпечення безпеки.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Для оцінки якості розробленого програмного забезпечення було обрано такі ключові метрики якості:

– безпека;

– надійність;

– підтримуваність;

– дублювання коду;

– глибина ієрархії вебсторінок.

Було використано статичний аналіз коду за допомогою платформи SonarCloud [31], яка перевіряє вихідний код на наявність помилок, потенційних вразливостей, недоліків у стилі написання тощо (рисунок 4.1).

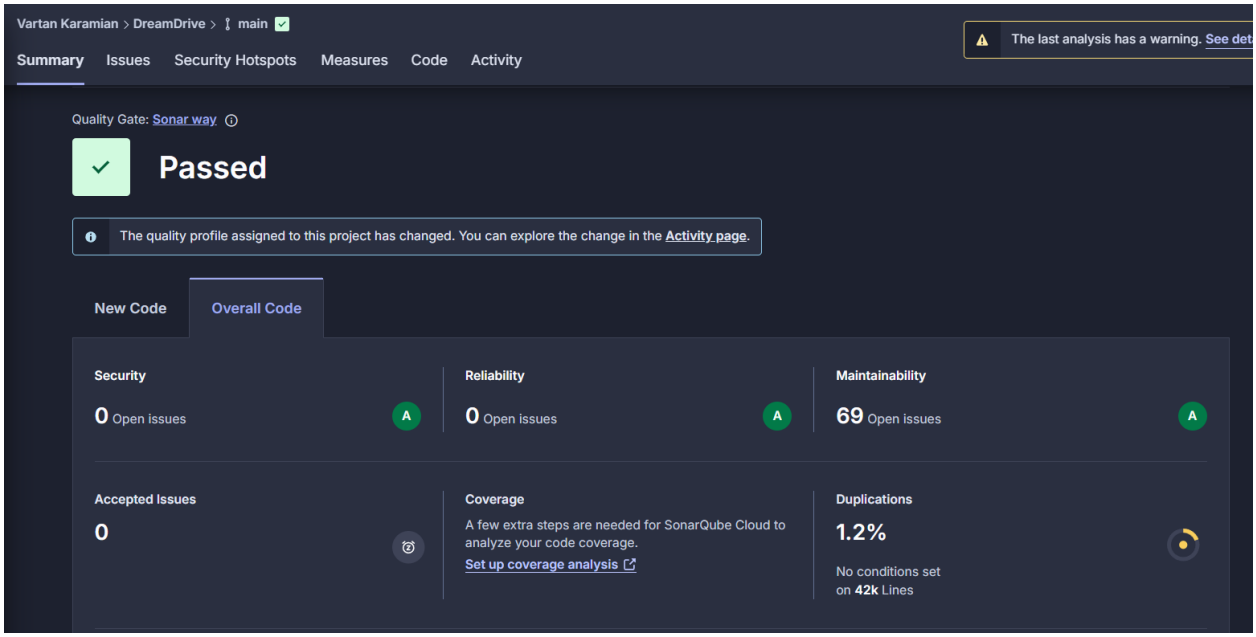


Рисунок 4.1 – Аналіз коду SonarCloud

У результаті аналізу було отримано позитивні результати, що свідчать про високу якість коду та відповідність основним нефункціональним вимогам.

За критерієм безпеки (Security) не виявлено жодних відкритих проблем чи вразливостей, а проєкт отримав найвищу оцінку A, що свідчить про відсутність ризиків, пов’язаних з можливими кіберзагрозами або порушенням

захисту даних. Аналогічно, за показником надійності (Reliability) також не зафіксовано жодних багів або критичних помилок у коді. Проєкт отримав оцінку A, що підтверджує його стабільну та передбачувану роботу без збоїв.

У категорії підтримуваності (Maintainability) система виявила 69 code smells, тобто ділянок коду, які можуть ускладнювати його підтримку або розвиток у майбутньому. Попри це, загальна оцінка залишилася на рівні A, що вказує на невисокий технічний борг та хорошу структурованість коду. Рівень дублювання коду (Duplications) склав 1.2% на загальну кількість понад 42 тисячі рядків, що вважається прийнятним показником.

Однією з метрик, що використовувалась для оцінки якості фронтенд-частини застосунку, є глибина ієрархії вебсторінок — тобто максимальна кількість переходів, які користувач повинен зробити від головної сторінки до найбільш вкладеного елемента інтерфейсу. У даному випадку цей показник дорівнює 4, що свідчить про помірну складність структури навігації. Така глибина вважається прийнятною для вебзастосунків із багатофункціональним інтерфейсом, оскільки забезпечує логічну організацію контенту без перевантаження користувача, проте важливо стежити, щоб подальше збільшення не знижувало зручність навігації.

Загалом, результати аналізу підтверджують, що програмне забезпечення відповідає високим стандартам якості, є безпечним, надійним і підтримуваним, зручним з точки зору досвіду користувача.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконано модульне та мануальне тестування програмного забезпечення. Підготовлено тест-кейси, які охоплюють основні функціональні модулі: головна сторінка, авторизація, управління користувачами, платежі, розклад, навчальні матеріали та тестування. Відповідні тест-кейси описані у таблицях 4.1 – 4.15.

Таблиця 4.1 – Тест 1.1 Перегляд головної сторінки

Початковий стан системи	Користувач неавторизований і відкриває головну сторінку сайту автошколи.
Вхідні дані	Відсутні.
Опис проведення тесту	Користувач відкриває головну сторінку сайту в браузері. Перевіряє наявність навігаційної панелі з доступними сторінками: «Головна», «Ціни», «Філії», «Про нас», «Відгуки», «Питання» та «Контакти». Користувач послідовно переходить на кожну з цих сторінок, перевіряючи коректність завантаження контенту. На сторінці «Ціни» користувач натискає кнопку «Обрати» під одним із тарифних планів, після чого система має перенаправити його на сторінку авторизації. У разі помилок завантаження компонентів має з’являтися відповідне повідомлення.
Очікуваний результат	Усі сторінки завантажуються коректно, контент відображається повністю, кнопка «Обрати» працює та перенаправляє на сторінку авторизації. У разі помилки відображається зрозуміле повідомлення.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.2 – Тест 2.1 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, ім’я, прізвище, пароль, підтвердження паролю.

Продовження таблиці 4.2

Опис проведення тесту	Користувач заповнює всі обов’язкові поля форми реєстрації, вводячи коректну електронну пошту, яка раніше не була зареєстрована в системі, а також ім’я та прізвище. У поле пароля вводиться надійний пароль довжиною від 10 до 64 символів, який містить хоча б одну англійську літеру, одну цифру та один спеціальний символ, і не входить до списку найпопулярніших паролів. У поле підтвердження пароля вводиться точно такий самий пароль. Після заповнення всіх полів користувач натискає кнопку підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему. Відображається повідомлення "Реєстрація успішна", після чого виконується перенаправлення на сторінку входу.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 2.2 Авторизація користувача

Початковий стан системи	Користувач знаходиться на сторінці входу в систему.
Вхідні дані	Електронна пошта, пароль.
Опис проведення тесту	Користувач вводить у відповідні поля коректну електронну пошту, що вже була зареєстрована в системі, та правильний пароль, що відповідає цьому акаунту. Після заповнення форми користувач натискає кнопку «Увійти». У разі правильного введення даних очікується виконання авторизації та перенаправлення до особистого кабінету користувача.

Продовження таблиці 4.3

Очікуваний результат	Авторизація проходить успішно, з’являється повідомлення про успішний вхід, і користувач перенаправляється на панель керування з доступом відповідно до своєї ролі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 2.3 Панель керування

Початковий стан системи	Користувач успішно авторизувався в системі.
Вхідні дані	Облікові дані користувача (ролі: студент, викладач, адміністратор).
Опис проведення тесту	Після авторизації користувач автоматично перенаправляється на панель керування. На екрані відображаються картки з короткою інформацією, яка відповідає ролі користувача. Студент бачить майбутні заняття, прогрес у навчанні та відгуки. Адміністратор або викладач бачить відповідні службові картки. Також користувач має доступ до основних розділів через картки: «Навчальні матеріали», «Платежі», «Тести», «Розклад», «Групи» тощо. Якщо користувач зі статусом "студент" не оплатив навчання, деякі функції заблоковані.
Очікуваний результат	Панель керування відображається відповідно до ролі користувача. Інформація актуальна. Картки доступу до функціональних модулів працюють. При неоплаченому навчанні частина функціональності заблокована та відображається повідомлення з пропозицією оплатити.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 3.1 Управління користувачами

Початковий стан системи	Користувач авторизований у системі з роллю адміністратора.
Вхідні дані	Відсутні
Опис проведення тесту	Адміністратор переходить до розділу керування користувачами через панель навігації. На екрані з'являється таблиця з переліком усіх зареєстрованих користувачів. Кожен запис у таблиці містить інформацію про ім’я, прізвище, електронну пошту, роль, статус оплати, статус активності та контактні дані. Адміністратор перевіряє наявність повного списку користувачів, а також коректність відображення інформації для кількох вибраних записів. У випадку, якщо користувачів не знайдено або сталася помилка завантаження даних, має з’явитися повідомлення про помилку.
Очікуваний результат	Таблиця користувачів завантажується успішно, інформація відображається коректно. У разі помилки відображається повідомлення про проблему з оновленням або відсутністю даних.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 3.2 Перегляд груп та студентів

Початковий стан системи	Користувач авторизований у системі під роллю викладача.
Вхідні дані	Обрано групу

Продовження таблиці 4.6

Опис проведення тесту	Викладач переходить до розділу “Мої групи” через панель навігації. Система відображає список навчальних груп, прикріплених до викладача. Викладач обирає одну з груп, після чого система відкриває список студентів цієї групи, де виводиться основна інформація: ПІБ, email, показники успішності. Викладач має змогу переглянути детальну інформацію кожного студента. Якщо у викладача немає жодної групи, система повідомляє “Немає закріплених груп”. Якщо у вибраній групі немає студентів — повідомлення “У цій групі ще немає студентів”.
Очікуваний результат	Викладач бачить список своїх навчальних груп. Після вибору групи викладач отримує список студентів із можливістю перегляду детальної інформації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 4.1 Оплата навчання

Початковий стан системи	Користувач авторизований у системі
Вхідні дані	Вибір тарифного плану для оплати.
Опис проведення тесту	Користувач переходить до розділу «Платежі» та обирає бажаний тарифний план. Після цього відбувається перенаправлення на платіжну платформу LiqPay, де користувач вводить реквізити банківської картки або обирає оплату через Google Pay чи Privat24. Після завершення оплати користувач повертається на сайт і отримує повідомлення про статус оплати. Далі користувач повертається на сторінку з платежами.

Продовження таблиці 4.7

Очікуваний результат	Оплата проходить успішно, статус оплати підтверджено, роль користувача змінюється на «студент», і користувач отримує повний доступ до функціоналу системи.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 4.2 Перегляд усіх платежів

Початковий стан системи	Адміністратор авторизований у системі
Вхідні дані	Відсутні
Опис проведення тесту	Адміністратор переходить до розділу «Платежі» або відповідного розділу огляду платежів у системі. Система відображає статистику по платежам та список усіх здійснених оплат з деталями: дата платежу, статус, ім’я студента, пакет послуг та сума. Адміністратор переглядає інформацію, перевіряє коректність відображених даних.
Очікуваний результат	Адміністратор успішно отримує доступ до повної історії платежів з усіма необхідними деталями.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 5.1 Перегляд навчальних матеріалів

Початковий стан системи	Користувач (студент або вчитель) авторизований у системі та знаходиться на головній панелі або в меню.
Вхідні дані	Відсутні

Продовження таблиці 4.9

Опис проведення тесту	Користувач переходить у розділ з навчальними матеріалами. Система відображає офіційні матеріали з правил дорожнього руху, поділені на три секції: правила дорожнього руху по розділах, дорожні знаки за групами та дорожню розмітку за групами. Користувач переглядає наявні матеріали та перевіряє їх структурованість і доступність.
Очікуваний результат	Користувач успішно отримує доступ до всіх навчальних матеріалів, може переглядати їх відповідно до своєї ролі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 5.2 Управління навчальними матеріалами

Початковий стан системи	Адміністратор авторизований у системі
Вхідні дані	Відсутні
Опис проведення тесту	Адміністратор заходить у розділ керування навчальними матеріалами, переглядає існуючі теми, правила, дорожні знаки та структуру матеріалів. Він додає нові записи або редагує наявні, прикріплює зображення, змінює текст правил. Після внесення змін натискає кнопку збереження. У разі пропуску обов’язкових полів або виникнення помилки система відображає відповідне повідомлення.
Очікуваний результат	Навчальні матеріали успішно додані або оновлені, зміни збережені, і матеріали стають доступними користувачам з відповідними правами.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.11 – Тест 6.1 Перегляд своїх занять

Початковий стан системи	Користувач авторизований у системі як викладач або студент
Вхідні дані	Відсутні
Опис проведення тесту	Користувач переходить у розділ «Мої заняття». Якщо це студент, він бачить список своїх запланованих та завершених практичних і теоретичних занять із відповідним статусом кожного. Якщо це вчитель, він бачить усі свої заплановані заняття. Якщо запланованих чи завершених занять немає, система відображає повідомлення «Занять немає».
Очікуваний результат	Користувач успішно переглядає актуальний список своїх майбутніх та минулих занять або отримує повідомлення про їх відсутність.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.12 – Тест 6.2 Бронювання практичних занять

Початковий стан системи	Користувач авторизований у системі як студент
Вхідні дані	Вибір дати, вибір заняття
Опис проведення тесту	Користувач переходить у розділ «Мої заняття» та обирає дату для перегляду практичних занять. Система відображає список доступних практичних занять на обрану дату. Користувач вибирає зручний час і натискає кнопку бронювання. Якщо у користувача немає активного бронювання, заняття успішно бронюється і відображається у списку як заброньоване. Якщо ж користувач уже має активне бронювання, система відображає повідомлення про те, що нове бронювання зробити неможливо.

Продовження таблиці 4.12

Очікуваний результат	Користувач успішно бронює практичне заняття, або у випадку наявності існуючого бронювання отримує відповідне повідомлення про неможливість зробити ще одне.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.13 – Тест 6.3 Інтерактивний календар

Початковий стан системи	Користувач авторизований у системі як викладач або адміністратор
Вхідні дані	Відсутні
Опис проведення тесту	Вчитель отримує доступ до відображення свого розкладу у вигляді інтерактивного календаря з подіями — запланованими заняттями. Календар можна переглядати у режимі поточного тижня або одного дня. Користувач може натиснути на заняття, щоб відредагувати або скасувати його, якщо до початку заняття залишилось не менше 24 годин. Також можна додати нове заняття, натиснувши на вільний таймслот і заповнивши форму. Адміністратор спочатку вибирає викладача, чиї заняття він хоче переглянути, після чого отримує аналогічний функціонал для управління розкладом. При будь-яких змінах система відображає спливаючі повідомлення зі статусом операції.
Очікуваний результат	Користувач успішно переглядає, додає, редагує або видаляє заняття в календарі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.14 – Тест 7.1 Проходження тестів

Початковий стан системи	Користувач авторизований у системі як студент
Вхідні дані	Вибір типу тесту (іспит, тест за темами або тест від викладача).
Опис проведення тесту	Користувач переходить у модуль «Інтерактивні тести» та обирає тип тестування. Система завантажує відповідний набір питань і запускає тест. Після проходження всіх питань користувач завершує тестування, після чого система відображає результати тесту і список допущених помилок. Якщо під час тестування виникають технічні проблеми або питання недоступні, система виводить повідомлення з порадою повторити спробу пізніше.
Очікуваний результат	Користувач успішно проходить тест і отримує коректний результат з можливістю перегляду помилок, або в разі технічних проблем отримує відповідне інформативне повідомлення.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.15 – Тест 7.2 Перегляд результатів тесту

Початковий стан системи	Користувач авторизований у системі як студент
Вхідні дані	Вибір конкретного пройденого тесту для перегляду деталей.
Опис проведення тесту	Користувач переходить у розділ «Результати тестів», обирає один із завершених тестів і відкриває деталі. Система відображає повний перелік питань тесту, відповіді студента, правильні відповіді для допущених помилок та короткі пояснення. Користувач може повернутися назад на сторінку списку тестів, натиснувши кнопку «До тестів».

Продовження таблиці 4.15

Очікуваний результат	Користувач отримує повну інформацію про результати тесту, включно з деталізацією помилок і поясненнями, а також можливість повернутися до списку тестів.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

Опишемо контрольний приклад з усіма розгалуженнями та особливостями кожної ролі. Почнемо з функціоналу доступного для всіх та відповідно користувача без ролі.

4.3.1 Неавторизований користувач

Спочатку користувачі потрапляють на головну сторінку автошколи. Навігаційна панель зверху містить посилання на декілька загальнодоступних сторінок, що дозволять ознайомитися з навчальним закладом (рисунок 4.2).

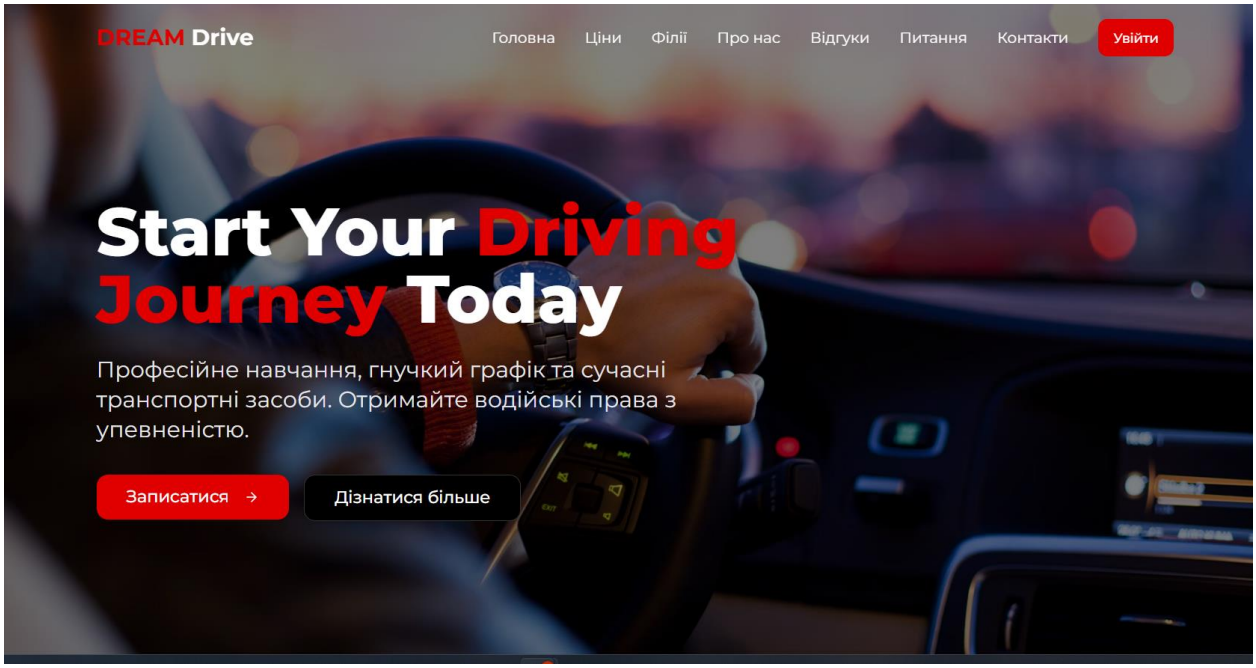


Рисунок 4.2 – Головна сторінка автошколи

Якщо натиснути кнопку «Записатися» або обрати в навігаційній панелі «Ціни», то перейдемо до сторінки «Ціни». Тут ми можемо обрати курс та натиснути «Придбати курс» (рисунок 4.3).

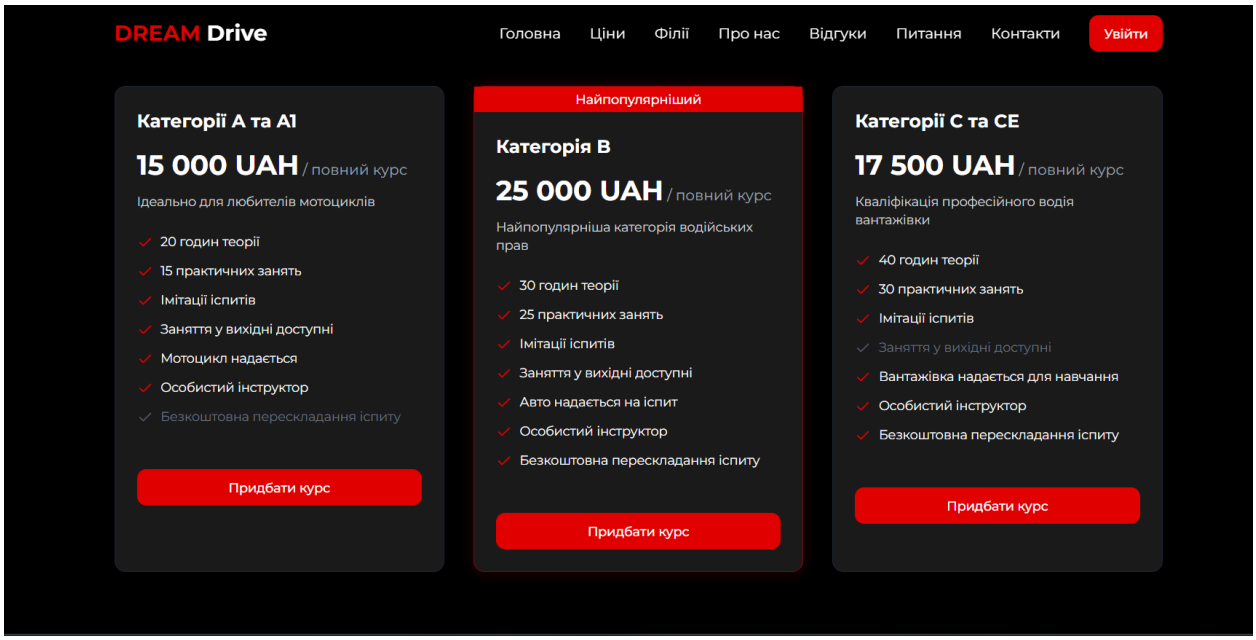


Рисунок 4.3 – Сторінка з цінами

Далі ми потрапляємо на сторінку реєстрації, де заповнюємо відповідні поля та натискаємо кнопку зареєструватися. Після чого отримуємо повідомлення з деталями помилки (рисунок 4.4).

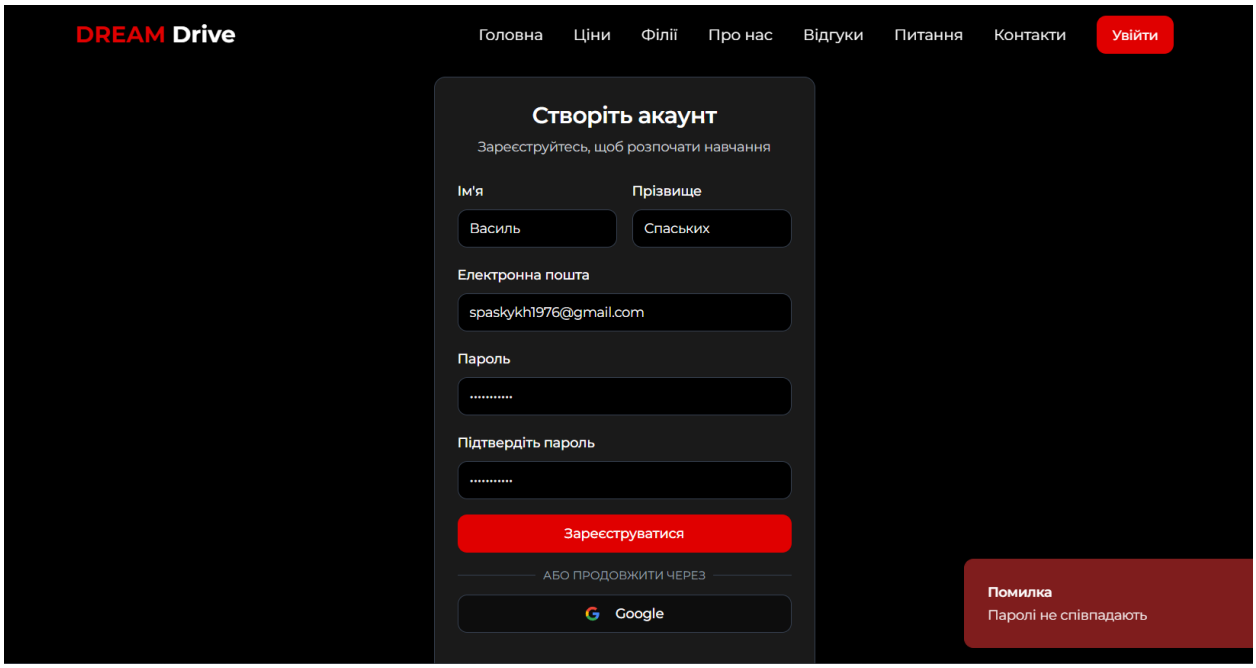


Рисунок 4.4 – Невдала спроба реєстрації

Відправимо форму ще раз, тільки з коректними даними – потрапляємо на панель керування з обмеженим доступом (рисунок 4.5).

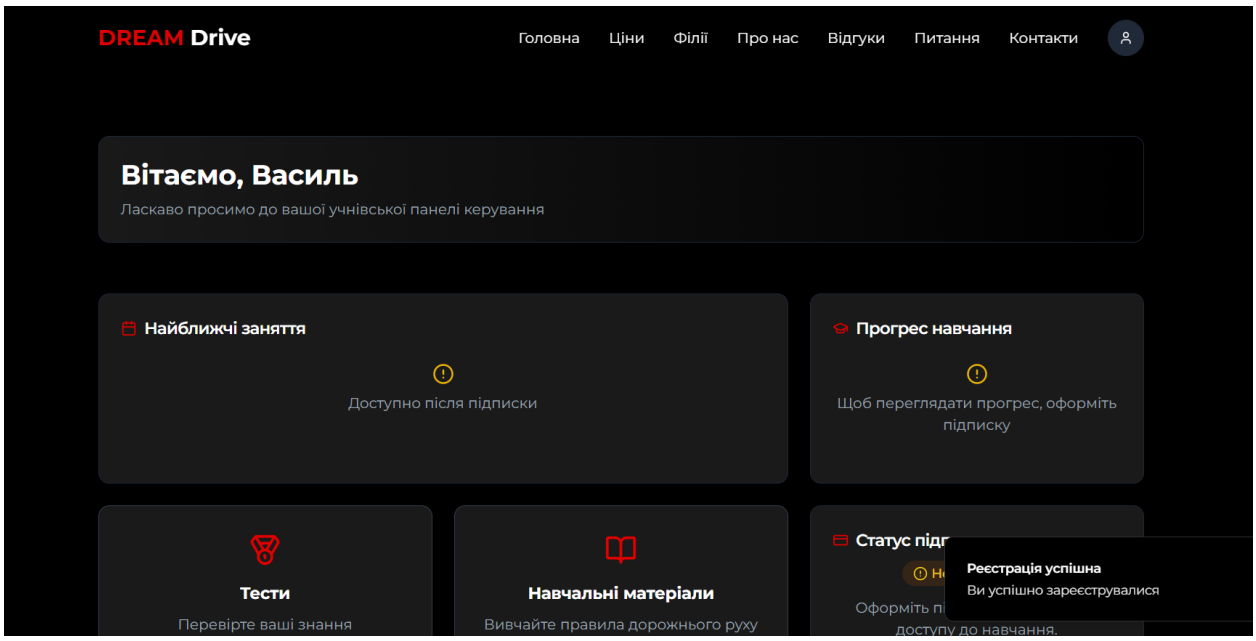


Рисунок 4.5 – Панель керування

Натиснувши на кнопку профілю та обравши секцію «Платежі», можемо продовжити оформлення оплати. Натискаємо «Оплатити» та переходимо до сервісу LiqPay. Заповнюємо платіжні реквізити та відправляємо платіж (рисунок 4.6).

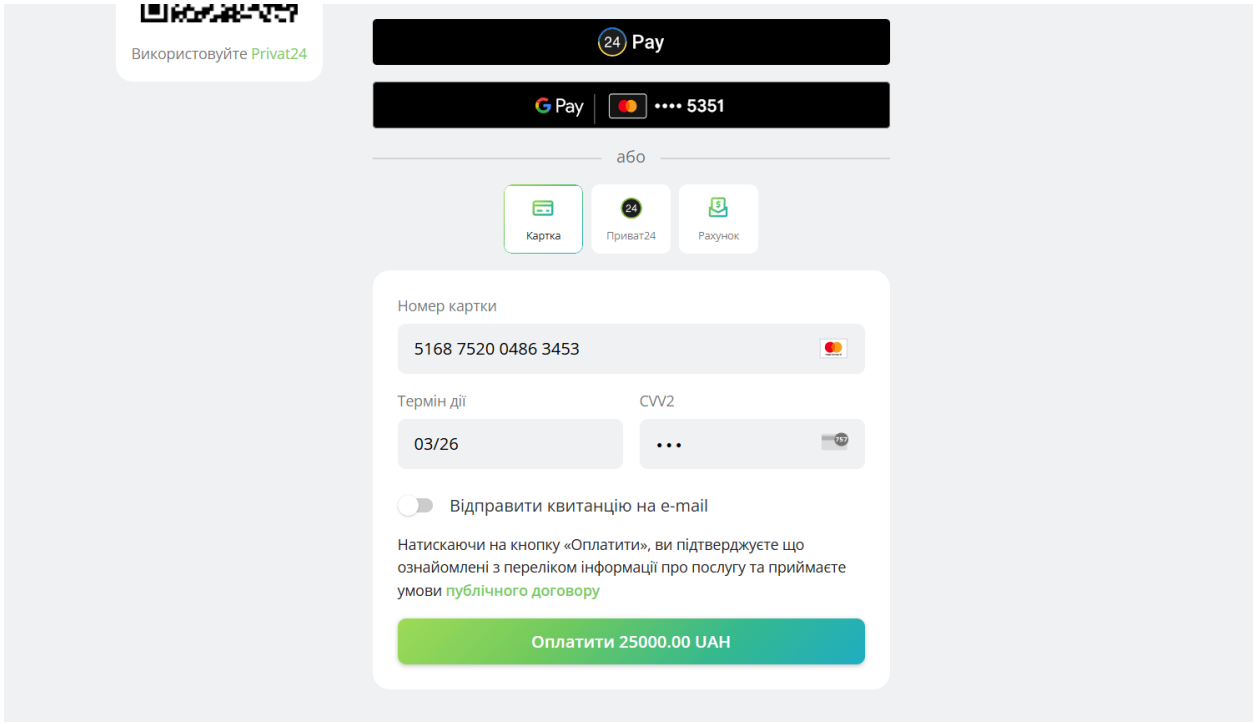


Рисунок 4.6 – Платіжний сервіс

Після успішної оплати отримуємо відповідне повідомлення та повертаємося на сторінку платежів, де можемо у цьому переконатися, рисунок 4.7.

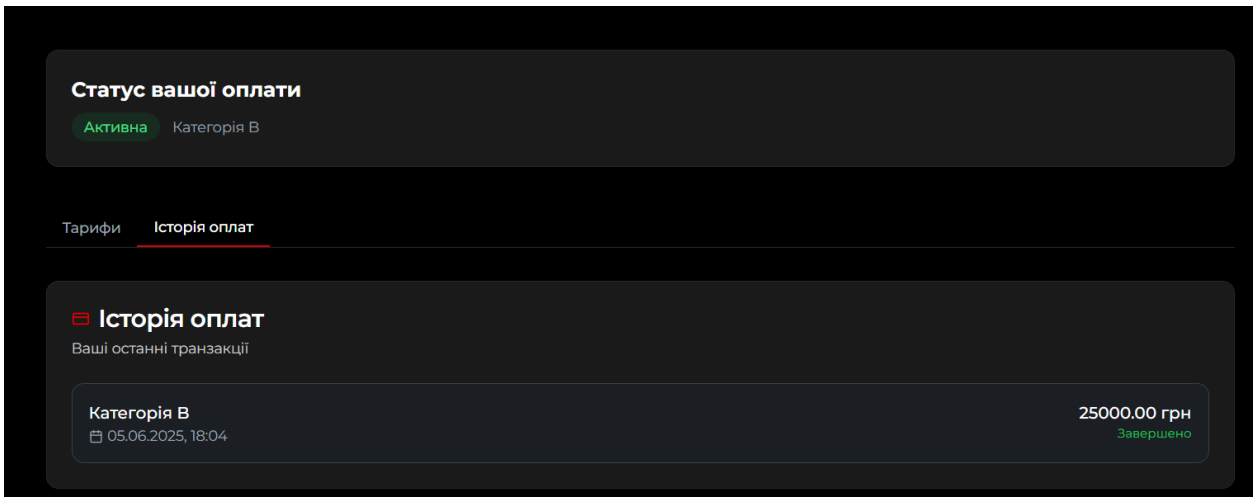


Рисунок 4.7 – Статус та історія оплат

4.3.2 Студент

Перейдемо на обліковий запис студента, який навчається в автошколі. При авторизації потрапляємо на панель керування (креслення 3). Перейдемо у вкладку «Мої заняття». Тут ми можемо переглянути усі заплановані заняття (рисунок 4.8) та забронювати практичне заняття на обрану дату (рисунок 4.9).

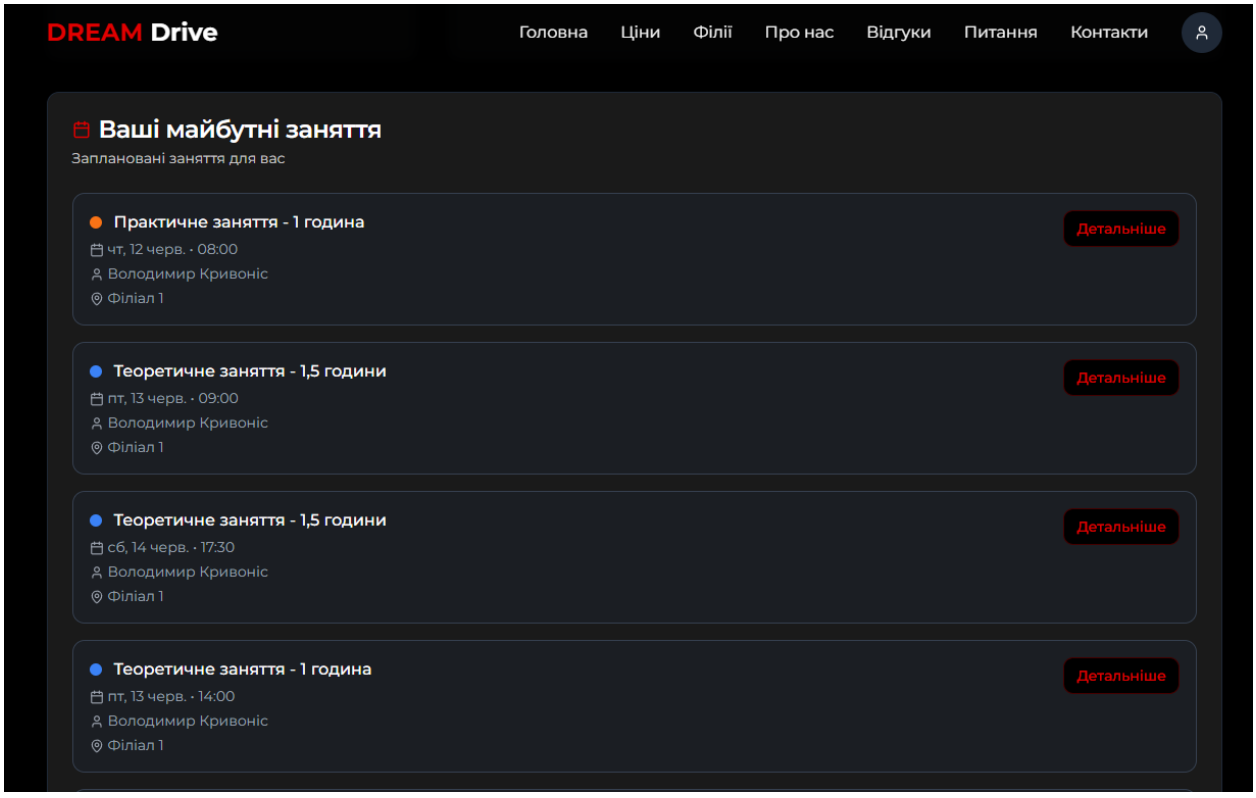


Рисунок 4.8 – Майбутні заняття

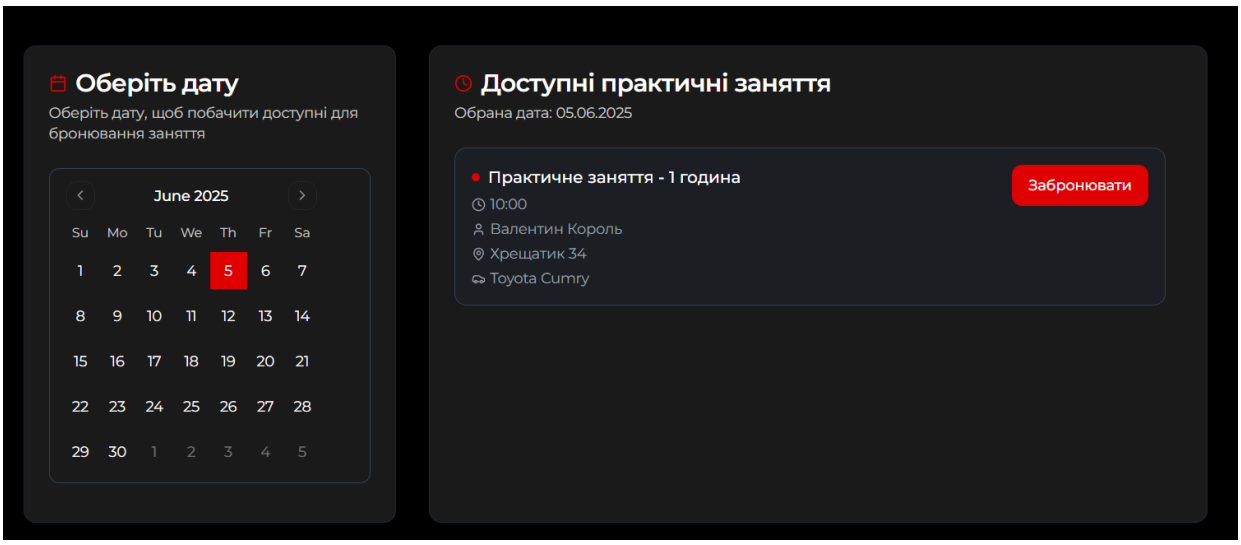


Рисунок 4.9 – Бронювання заняття

Студент може переглянути навчальні матеріали, які розділені на три секції та всередині секцій на розділи (креслення 3).

У модулі «Інтерактивні тести» потрібно обрати один з трьох видів тестів та почати іспит. Трохи нижче можна переглянути історію тестів та відкрити деталі тесту або повну історію (рисунок 4.10).

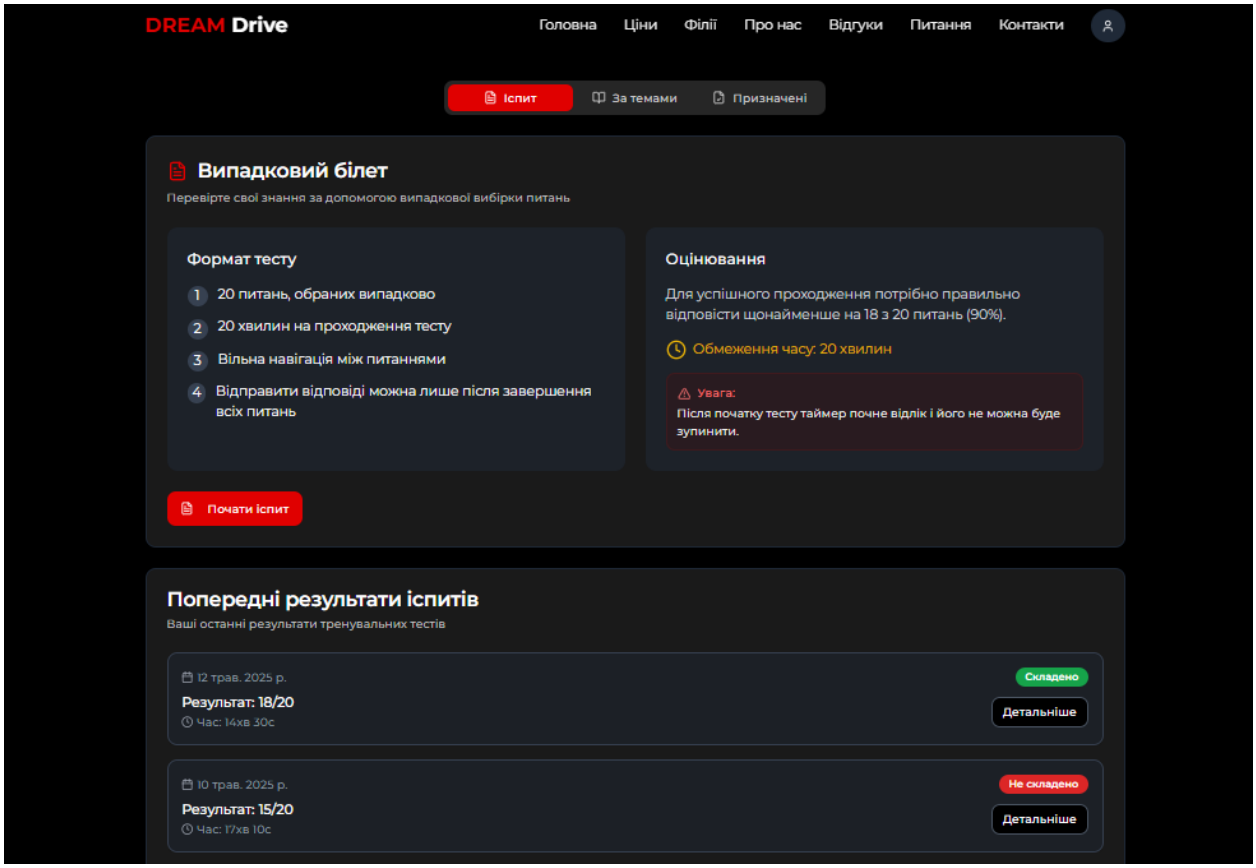


Рисунок 4.10 – Інтерактивні тести

Якщо почати іспит, то потрапляємо у вкладку з інформацією про тест та можемо його розпочати (рисунок 4.11). Далі послідовно відповідаємо на питання (креслення 3) та натискаємо «Завершити тест». Отримуємо результат тесту та нижче детальний розбір помилок (рисунок 4.12).

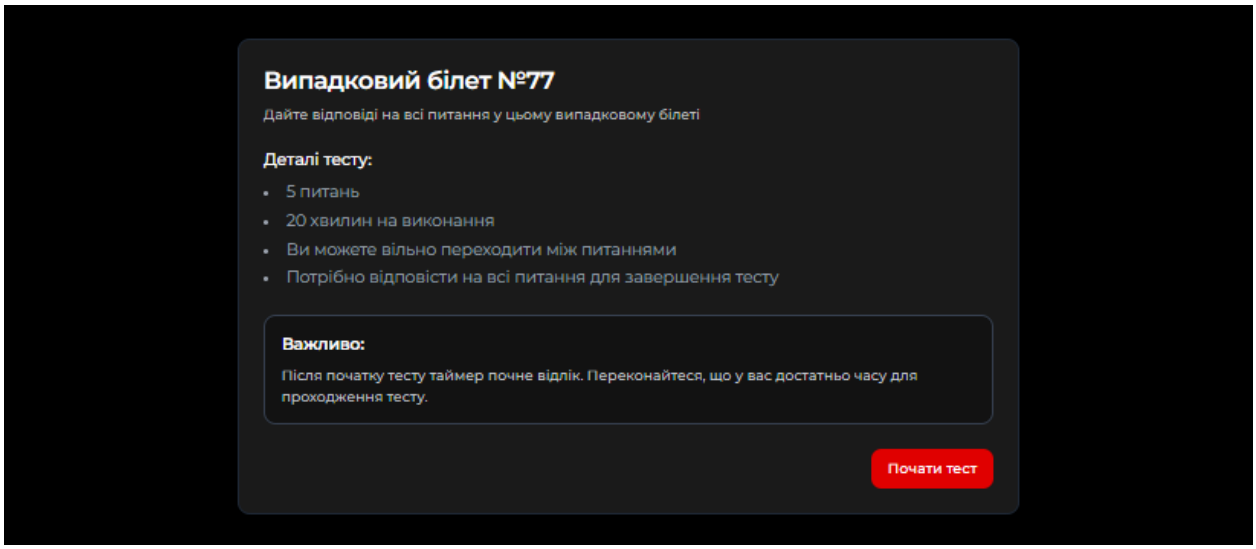


Рисунок 4.11 – Початок тесту

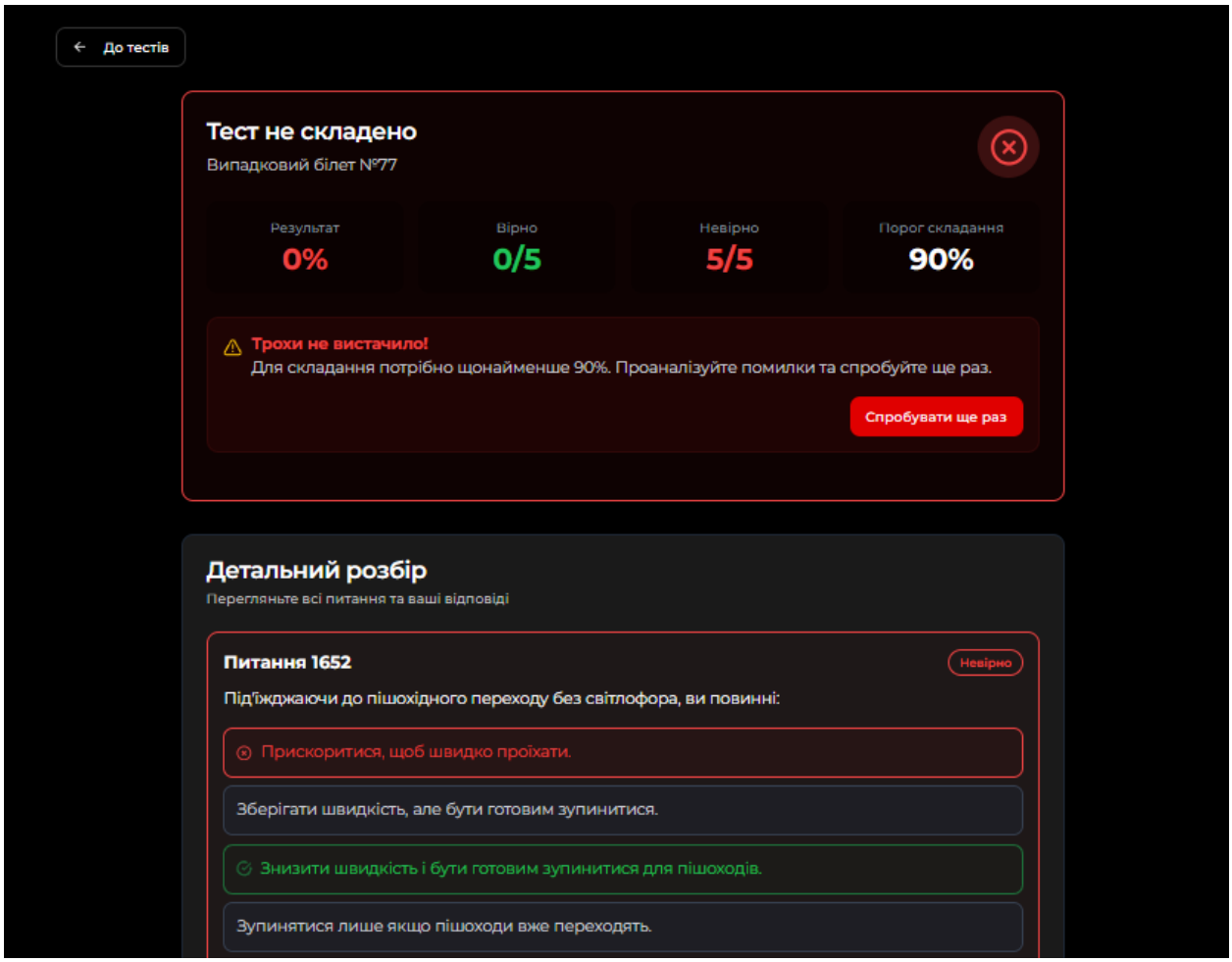


Рисунок 4.12 – Результат тесту

4.3.3 Викладач

Перейдемо на обліковий запис викладача (рисунок 4.13).

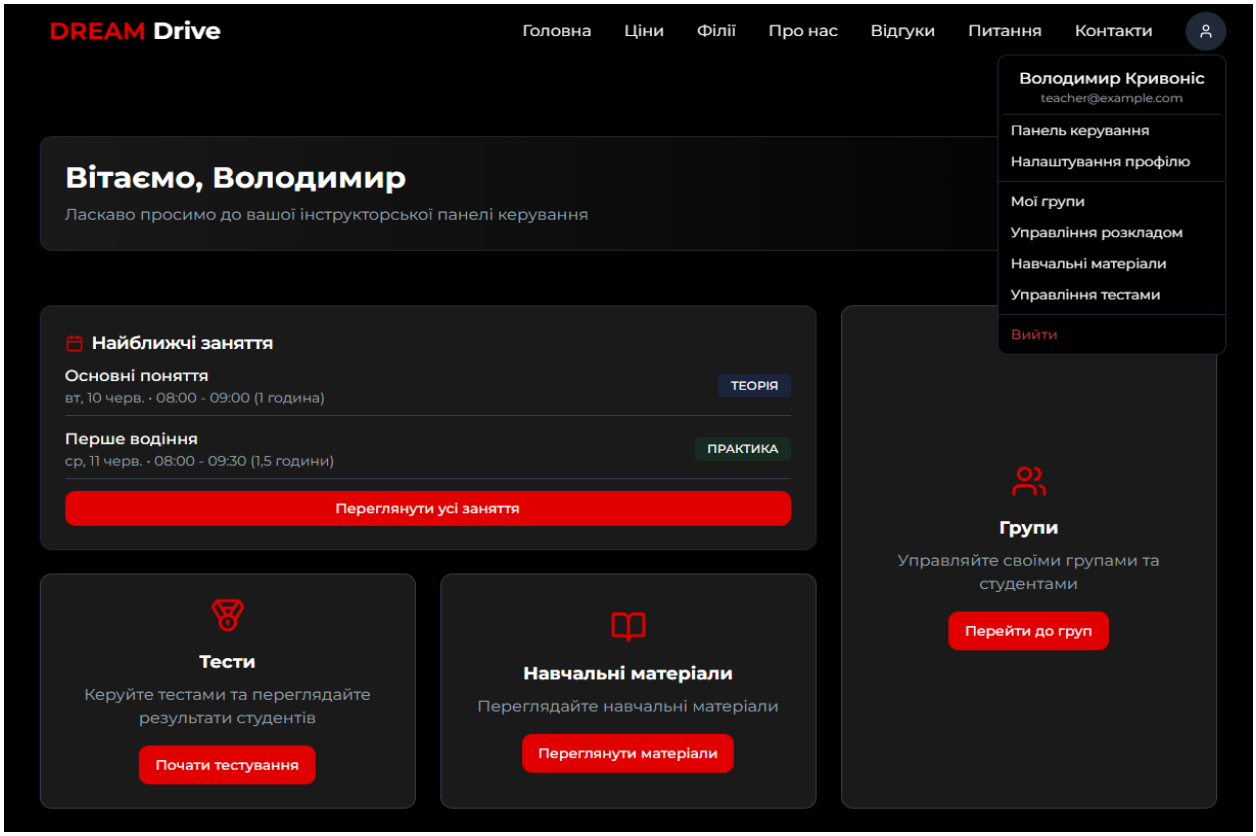


Рисунок 4.13 – Панель керування викладача

Перейдемо до груп, тут можна переглядати студентів (рисунок 4.14)

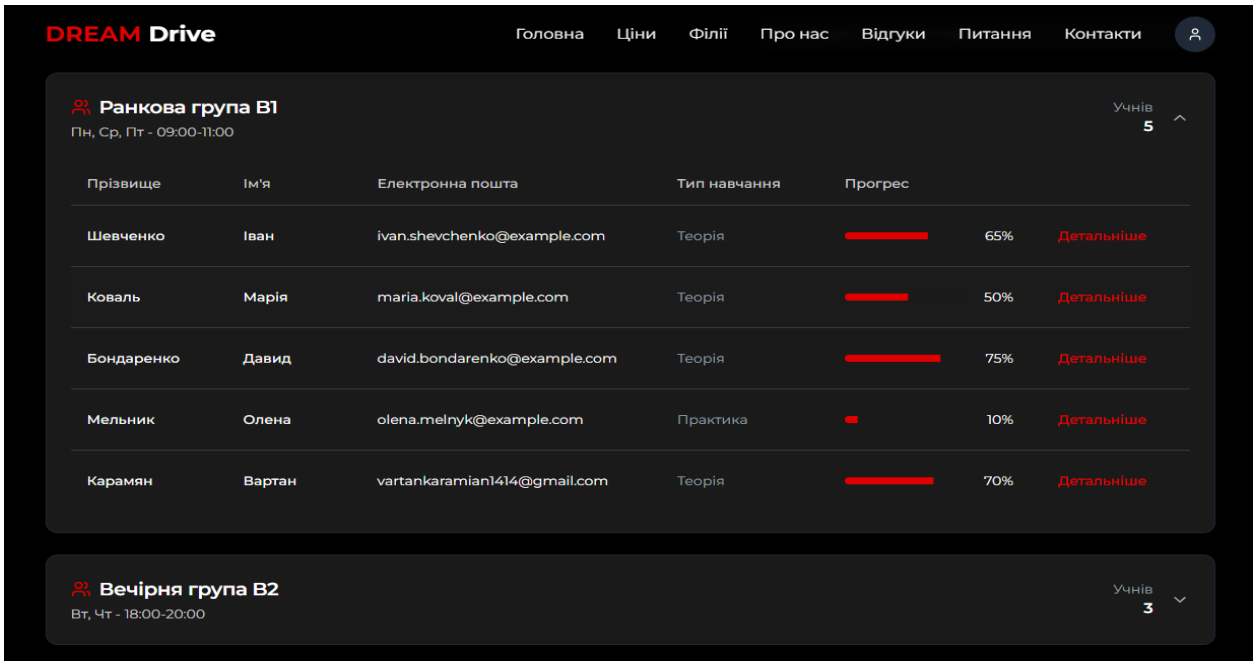


Рисунок 4.14 – Групи та студенти викладача

У вкладці управління розкладом можемо побачити інтерактивний календар занять (креслення 3). Якщо натиснути на вільний слот, можна додати нове заняття, заповнивши форму (рисунок 4.15). Натиснувши на заняття, можемо отримати детальну інформацію та відредагувати або відмінити його (рисунок 4.16).

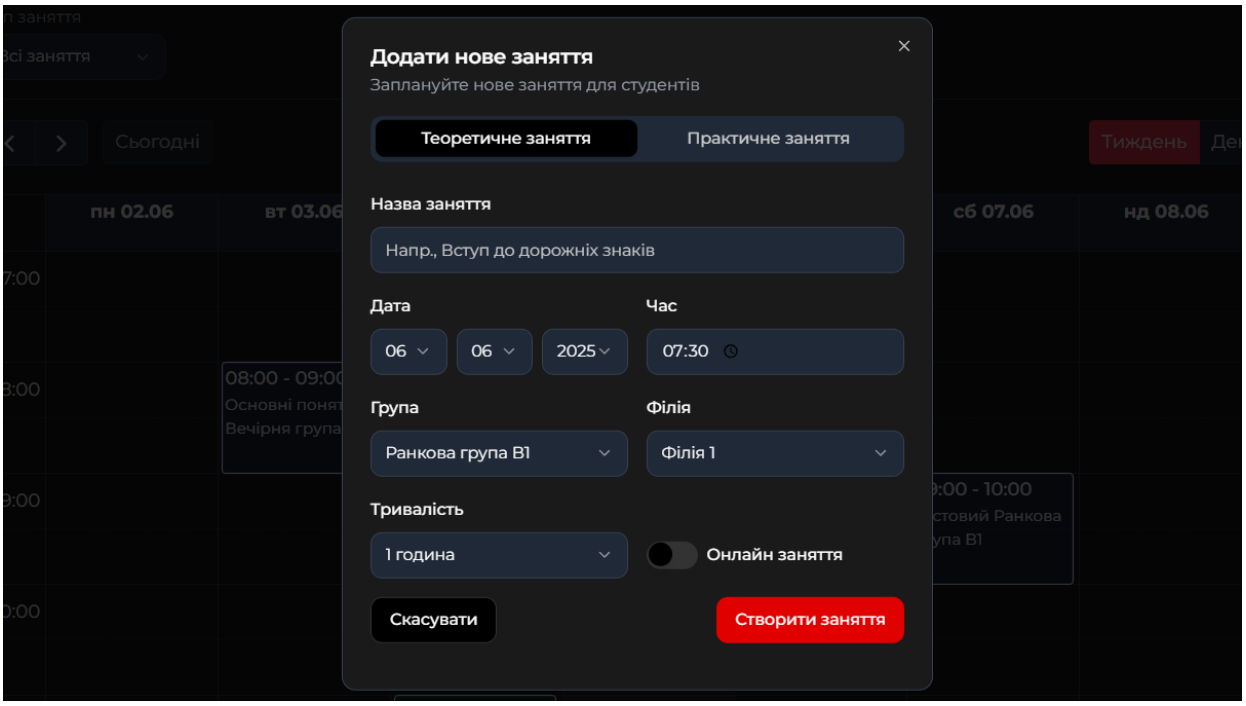


Рисунок 4.15 – Створення заняття

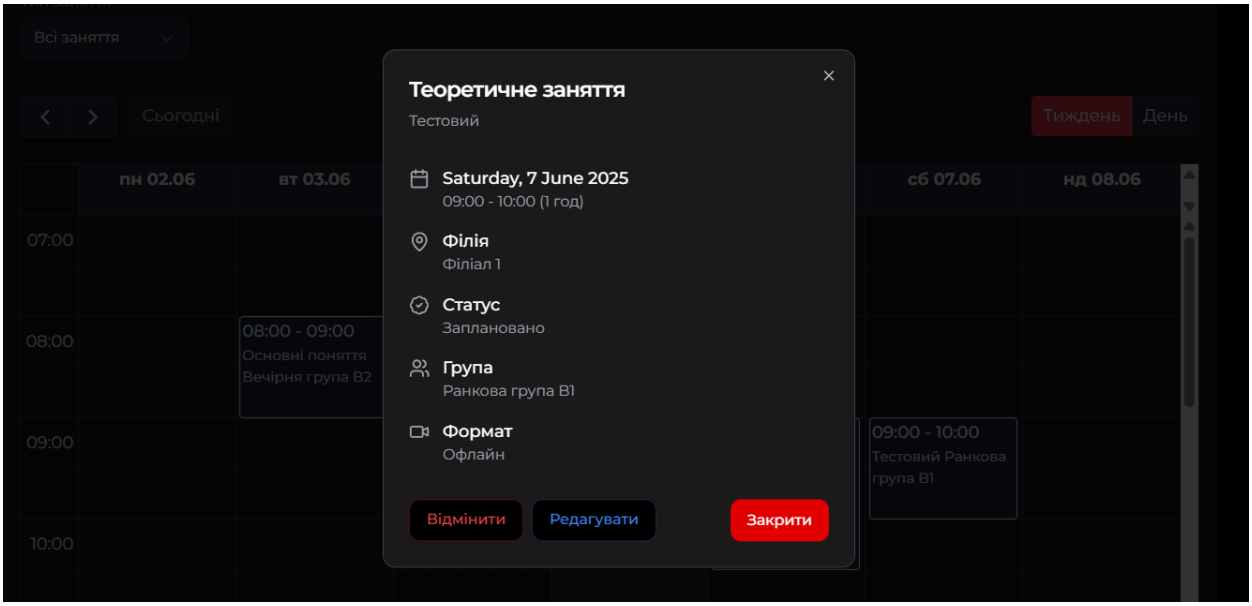


Рисунок 4.16 – Інформація про заняття

Якщо перейдемо до управління тестами. Тут можна переглянути опубліковані тести та їх статистику (рисунок 417). Можемо також створити новий тест: потрібно заповнити форму з деталями тесту, обрати питання з офіційного пулу та зберегти тест (рисунок 4.18).

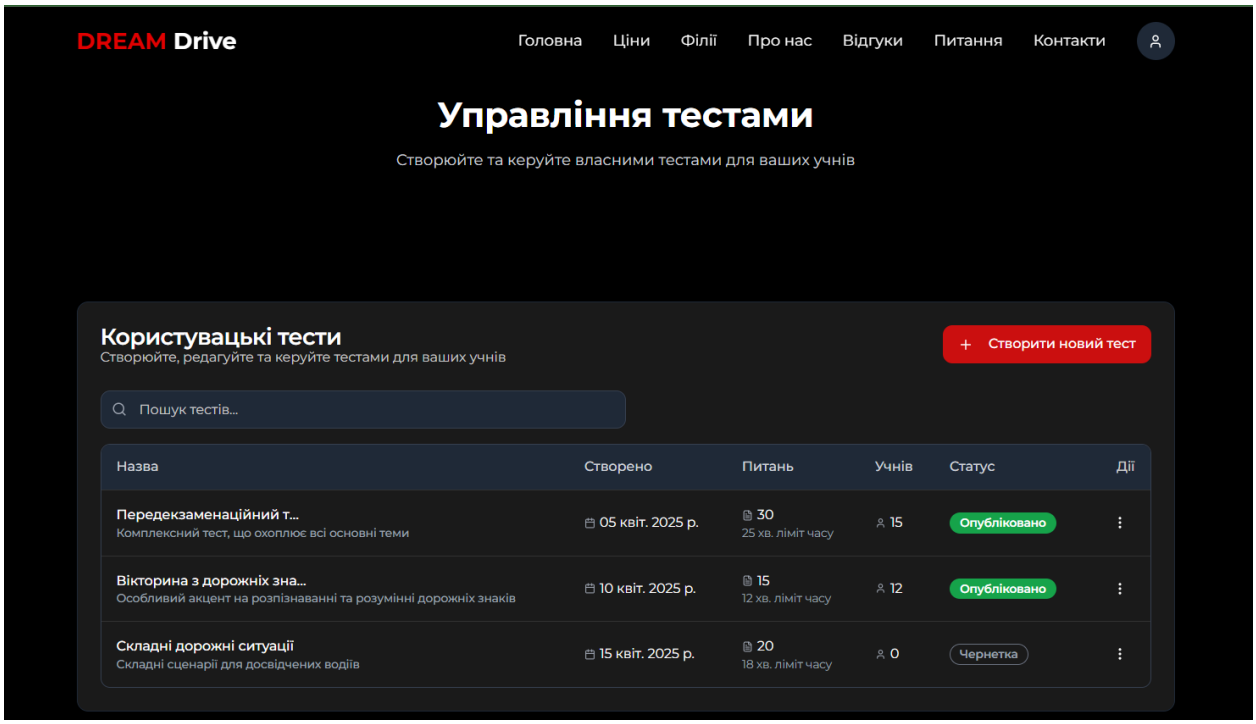


Рисунок 4.17 – Управління тестами

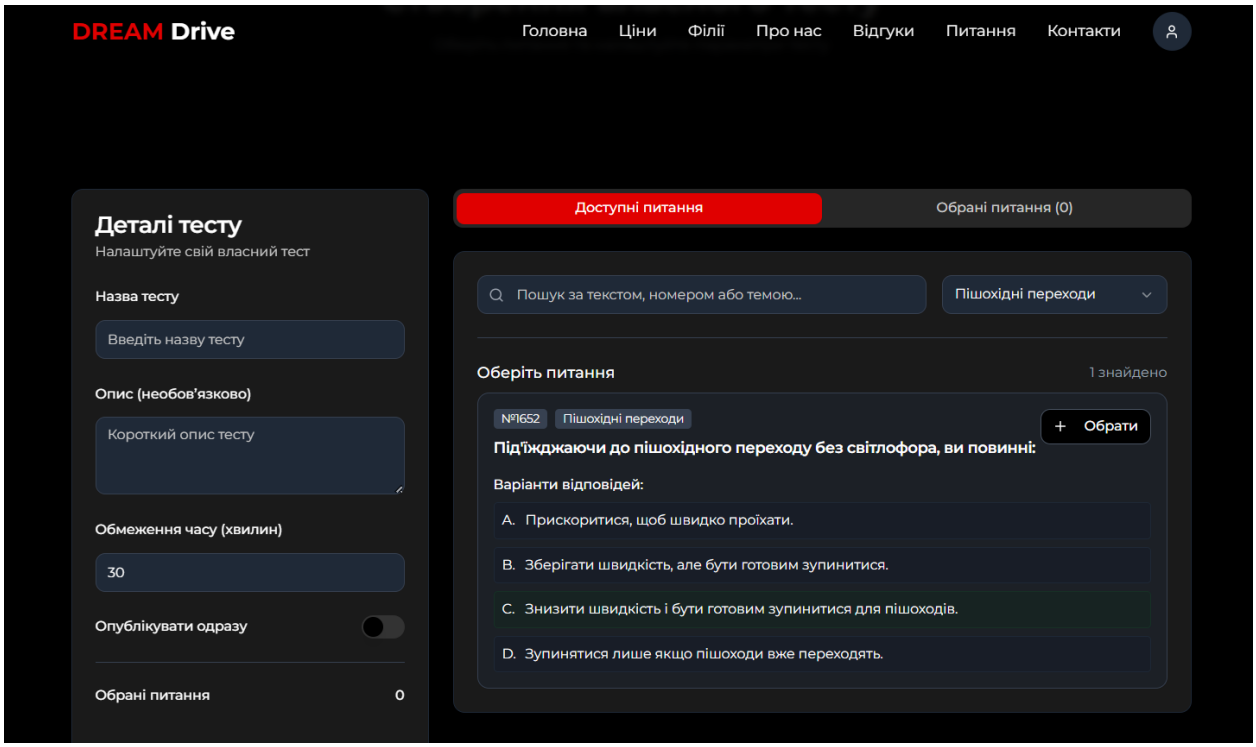


Рисунок 4.18 – Створення тесту

4.3.4 Адміністратор

Перейдемо на обліковий запис адміністратора. На панелі керування можна переглянути коротку статистику по автошколі та здійснити навігацію на інші сторінки (рисунок 4.19).

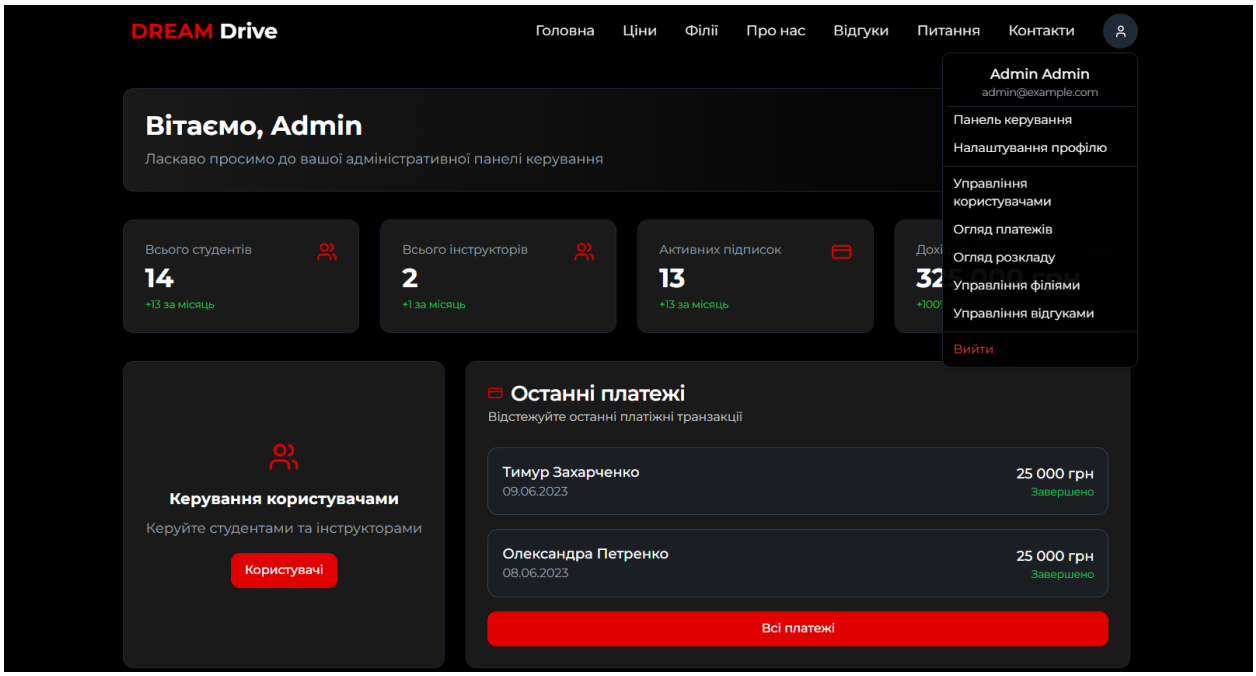


Рисунок 4.19 – Панель керування адміністратора

Адміністратор може переглядати платежі (рисунок 4.20).

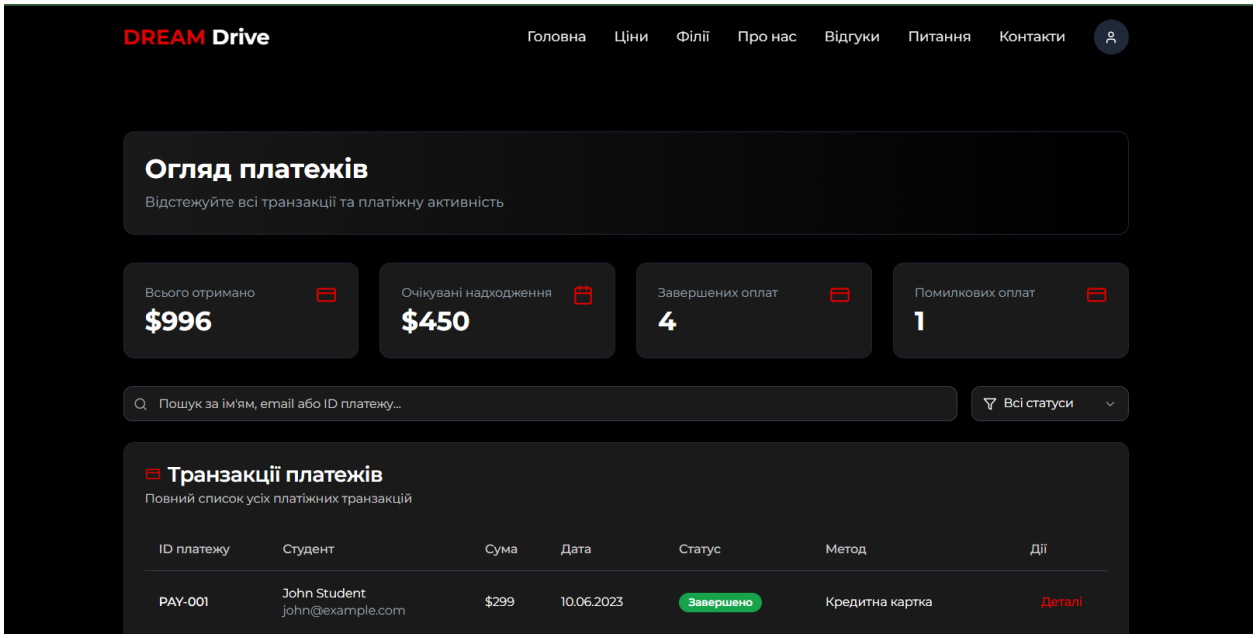


Рисунок 4.20 – Огляд платежів

Доступний перегляд календарю у розрізі кожного викладача (рисунок 4.21). Сторінки для управління користувачами (рисунок 4.22), управління відгуками (рисунок 4.23).

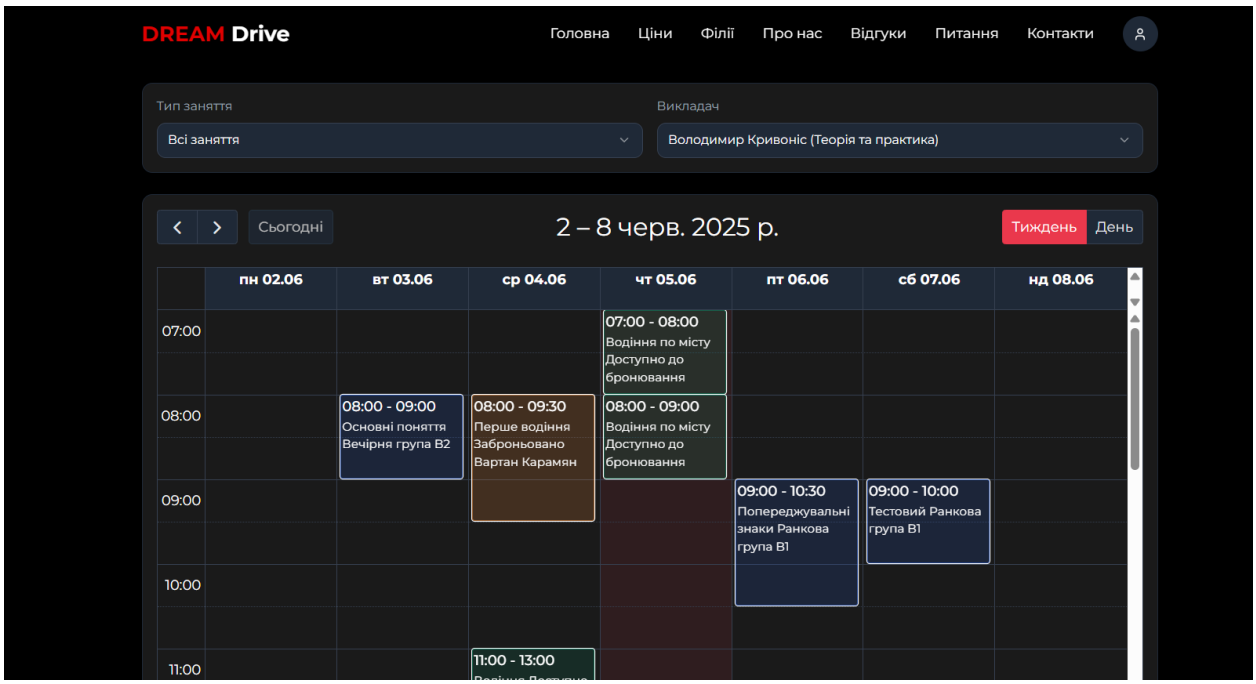


Рисунок 4.21 – Інтерактивний календар адміністратора

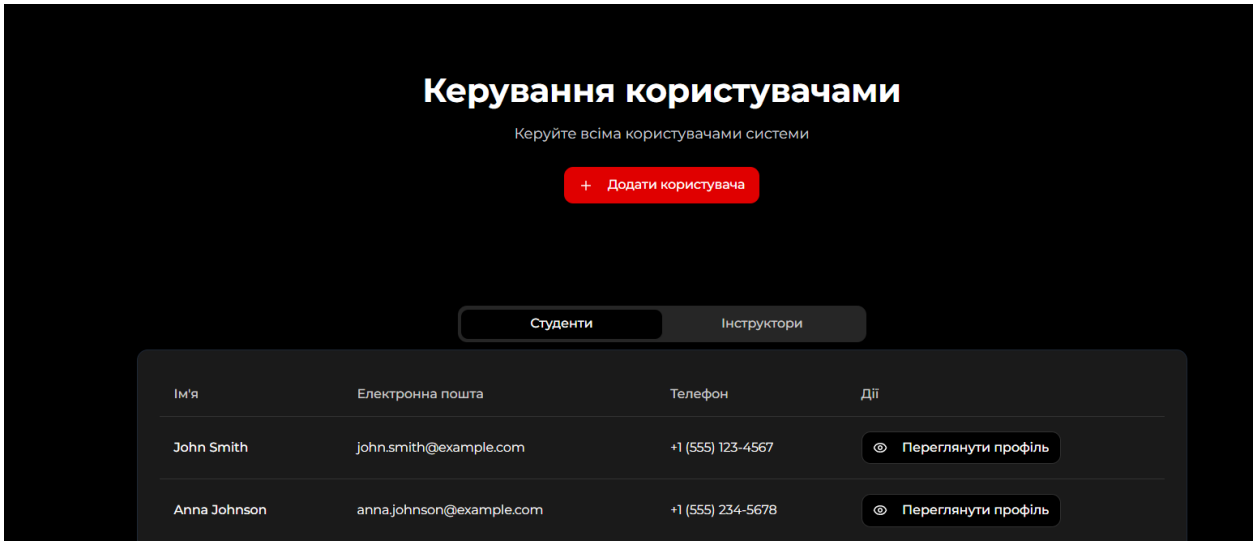


Рисунок 4.22 – Управління користувачами

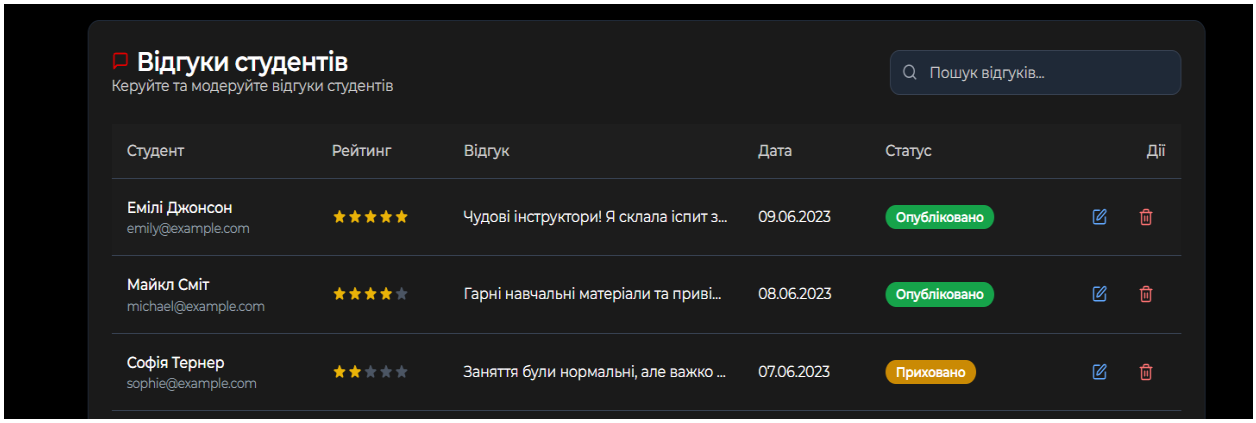


Рисунок 4.23 – Управління відгуками

Висновки до розділу

Отже, було проведено оцінку якості та тестування вебзастосунку. Для перевірки якості ПЗ було використано платформу SonarCloud, що дозволила дослідити вихідний код за ключовими метриками: безпека, надійність, підтримуваність, дублювання коду. Отримані результати свідчать про високу якість програмного забезпечення. Оцінки» за безпеку та підтверджують відсутність критичних вразливостей і багів. Підтримуваність коду залишилася на високому рівні попри незначну кількість code smells. Рівень дублювання коду склав лише 1.2% при понад 42 тис. рядків коду, що є допустимим значенням. Глибина навігаційної структури становить 4 рівні, що відповідає нормам для багатофункціонального інтерфейсу.

Було розроблено тест-кейси, які охоплюють усі основні функціональні модулі: головна сторінка, авторизація, управління користувачами, оплата, розклад, навчальні матеріали та тести. Окрему увагу приділено контрольному прикладу, який демонструє повний сценарій взаємодії користувача із системою залежно від його ролі — неавторизованого користувача, студента, викладача та адміністратора. Для кожної ролі було протестовано специфічні функції: реєстрація та оплата, перегляд та бронювання занять, доступ до навчальних матеріалів і тестів, створення занять і контроль тестів, керування користувачами та фінансовою інформацією.

Результати підтверджують, що система працює коректно, відповідає функціональним та нефункціональним вимогам, а її архітектура дозволяє ефективно масштабувати та підтримувати проєкт у майбутньому.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для ізоляції середовищ виконання кожного мікросервісу та зручного розгортання всього застосунку було використано Docker. Завдяки цьому кожен сервіс працює в окремому контейнері з власними залежностями та базою даних.

Для кожного з п’яти основних мікросервісів (AuthUser, Learning, Schedule, Payment, Public) використовується окремий Dockerfile (рисунок 5.1), який міститься у відповідній директорії сервісу.

Dockerfile × docker-compose.yml

```
FROM python:3.11-alpine

ENV PYTHONUNBUFFERED=1

COPY requirements.txt /tmp/requirements.txt
COPY requirements.dev.txt /tmp/requirements.dev.txt
COPY app /app
WORKDIR /app
EXPOSE 8000

ARG DEV=false
RUN python -m venv /py && \
    /py/bin/pip install --upgrade pip && \
    apk add --update --no-cache postgresql-client && \
    apk add --update --no-cache --virtual .tmp-build-deps \
        build-base \
        postgresql-dev \
        musl-dev && \
    /py/bin/pip install -r /tmp/requirements.txt && \
    if  [ "$DEV" = true ]; then \
```

Рисунок 5.1 – Фрагмент Dockerfile

Всі Dockerfile мають подібну будову: вони базуються на легкому образі python:3.11-alpine, встановлюють необхідні залежності з requirements.txt та requirements.dev.txt, додають користувача для безпечного запуску застосунку, налаштовують часову зону на Europe/Kyiv, і на завершення задають змінну середовища PATH для запуску через віртуальне середовище .

Основна конфігурація контейнерів відбувається через файл docker-compose.yml (рисунок 5.2), який знаходиться в корені репозиторію. У ньому

визначено сервіси для кожного з мікросервісів: окремо контейнер для самого застосунку та окремо для бази даних PostgreSQL. Для кожної бази даних створено іменований Docker-том, що зберігає дані між перезапусками. Запуск застосунків відбувається після перевірки доступності бази даних за допомогою спеціальної команди python manage.py wait_for_db.

Крім основних сервісів, у docker-compose.yml також визначено API Gateway на базі FastAPI, що виступає єдиною точкою входу для взаємодії з фронтендом, а також окремий контейнер для React-фронтенду. Кожен сервіс експонується на власному порту, що полегшує налагодження під час розробки.

Dockerfile | docker-compose.yml

```yaml
services:
  auth_user_service_app:
    container_name: auth_user_service-app-1
    build:
      context: ./backend/auth_user_service
      dockerfile: Dockerfile
      args:
        - DEV=true
    ports:
      - "8000:8000"
    volumes:
      - ./backend/auth_user_service/app:/app
    command: >
      sh -c "python manage.py wait_for_db &&
             python manage.py runserver 0.0.0.0:8000"
    env_file:
      - ./backend/auth_user_service/.env
    depends_on:
      - auth_user_service_db

  auth_user_service_db:
    image: postgres:15-alpine
    container_name: auth_user_service-db-1
    ports:
      - "5432:5432"
    volumes:
      - dev-db-data:/var/lib/postgresql/data
```

Рисунок 5.2 – Фрагмент docker-compose.yml

Повний набір сервісів docker-compose можна побачити у таблиці 5.1

Таблиця 5.1 – Опис сервісів

Сервіс	Порт	БД	Технологія
Auth Service	8000	PostgreSQL:5432	Django
Learning Service	8001	PostgreSQL:5433	Django
Schedule Service	8002	PostgreSQL:5434	Django
Payment Service	8003	PostgreSQL:5435	Django

Продовження таблиці 5.1

Сервіс	Порт	БД	Технологія
Public Service	8004	PostgreSQL:5436	Django
API Gateway	8005	-	FastAPI
Frontend	8080	-	React, Vite

Для безпосердньо розгортання бекенд-сервісів було обрано платформу Railway, оскільки вона надає простий і зручний інтерфейс для управління контейнерами та базами даних, що значно прискорює процес деплою і налаштування інфраструктури. Railway підтримує автоматичне масштабування, інтеграцію з GitHub, а також дозволяє легко керувати змінними середовища, що особливо важливо при роботі з кількома мікросервісами.

Створюємо акаунт на Railway та підключаємо репозиторій. Для кожного бекенд-сервісу створюємо окремий проєкт у Railway. У налаштуваннях проєкту вказуємо, що деплой має відбуватись через Dockerfile, а також підключаємо керовану PostgreSQL базу даних, яку пропонує Railway. У середовищі змінних вказуємо необхідні параметри підключення до бази та інші конфігурації. Після цього Railway автоматично збудує Docker-образ, запустить контейнер і надасть URL для кожного сервісу. Таким чином, ми отримуємо окремі бекенд-сервіси, кожен зі своєю базою і окремим доменом.

Для фронтенду ми використовуємо Vercel [32], який ідеально підходить для React + Vite проєктів. Створюємо акаунт на Vercel і підключаємо репозиторій фронтенду з GitHub. Vercel автоматично виявляє, що це React-проєкт, і пропонує стандартні налаштування для збірки та деплою.

У налаштуваннях проєкту на Vercel задаємо змінну середовища для адреси бекенд-сервісу API Gateway, яку ми отримали від Railway. Це дозволить фронтенду спілкуватися з уже розгорнутим бекендом. Після цього

Vercel виконує збірку проєкту і розміщує його на своєму CDN, надаючи публічний URL для доступу.

Візуалізуємо дану архітектуру за допомогою діаграми розгортання (рисунок 5.3). Інтерфейс користувача розміщується на Vercel та взаємодіє з API Gateway, розміщеним на окремому сервері. API Gateway спрямовує запити до відповідних мікросервісів. Кожен мікросервіс має власну базу даних для забезпечення ізоляції та незалежності компонентів.

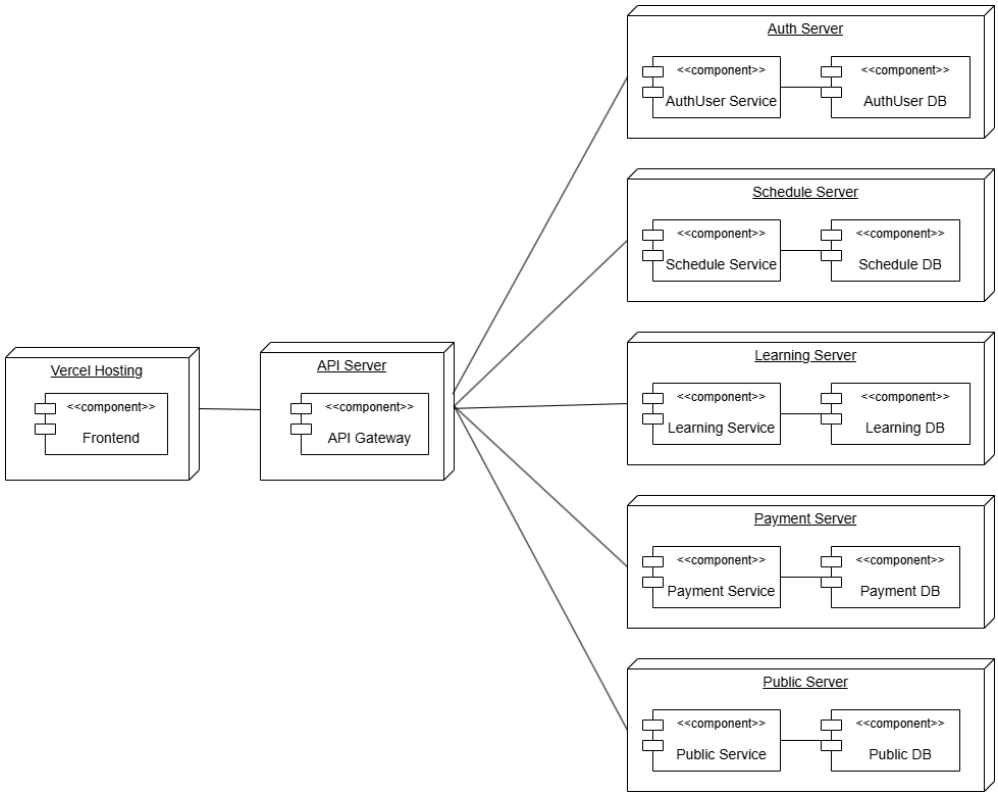


Рисунок 5.3 – Діаграма розгортання

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі «Керівництво користувача».

Супровід вебзастосунку DreamDrive реалізується з урахуванням вимог до безперервного доступу користувачів та стабільної роботи системи. Для цього впроваджено автоматизовані механізми оновлення як клієнтської частини, так і серверної логіки за допомогою сучасних хмарних платформ — Vercel для фронтенду та Railway для кожного окремого мікросервісу бекенду.

Усі зміни в коді проходять через систему контролю версій GitHub. Після перевірки і злиття оновлень у головну гілку (main), нова версія застосунку автоматично збирається та розгортається. Користувачам не потрібно виконувати жодних дій для оновлення: нова версія інтерфейсу одразу доступна у браузері після оновлення сторінки, а зміни на боці сервера починають діяти миттєво після успішного деплою Railway.

Фронтенд DreamDrive, побудований на React та Vite, розгортається на платформі Vercel, яка автоматично відстежує коміти до GitHub-репозиторію. Після кожної зміни система автоматично перезбирає проєкт і замінює попередню версію на нову.

Отже, DreamDrive не вимагає ручного встановлення оновлень ні від користувачів, ні від адміністратора. Усі оновлення виконуються безперервно, автоматично і без простою, що гарантує постійну доступність сервісу та актуальність функціональності.

Висновки до розділу

У розділі було описано процес розгортання та супроводу вебзастосунку DreamDrive. Для запуску мікросервісів використано Docker-контейнери, що дозволяє забезпечити ізоляцію, повторюваність середовищ і зручне масштабування. Розміщення серверної частини здійснюється на платформі Railway, де кожен сервіс має окремий проєкт і базу даних. Фронтенд розгортається на Vercel, яка автоматично виконує збірку проєкту після кожного оновлення репозиторію.

Супровід системи реалізується за допомогою GitHub та інтеграцій із хмарними платформами. Після злиття змін у гілку main, оновлення автоматично застосовується як для фронтенду, так і для кожного мікросервісу. Користувачам не потрібно встановлювати оновлення вручну — нові версії застосунку доступні одразу після публікації, що забезпечує постійну доступність і актуальність сервісу.

ВИСНОВКИ

У ході виконання дипломного проєкту розроблено вебзастосунок DreamDrive, який покриває основні бізнес-процеси автошколи. У результаті було зменшено кількість застосунків, потрібних для функціонування навчального закладу, з трьох до одного. Основні функції інтегровано в єдину систему, що підвищило ефективність роботи адміністрації та викладачів, а також навчання студентів.

Виконано усі поставлені функціональні задачі та вимоги. Розроблено головну сторінку та інтерефейс для її налаштування. Додано систему авторизації, яка реалізує гнучку системи ролей та прав. Для фінансових операцій впроваджено модуль платежів з інтеграцією LiqPay, завдяки якому студенти можуть здійснювати безпечні транзакції й переглядати історію оплат. Модуль навчальних матеріалів надає студентам доступ до ПДР, а адміністратор отримали інструменти для створення й редагування контенту. Розклад занять налаштовано таким чином, що студенти можуть переглядати й бронювати заняття, а викладачі та адміністратор використовують інтерактивний календар для управління заняттями. Нарешті, модуль запущено тестування, і студенти можуть проходити як офіційні, так і тести від викладачів. В той час як викладачі мають змогу створювати завдання, аналізувати результати та переглядати статистику успішності. Згідно з поставленими вимогами, всі функціональні блоки були інтегровані в єдину систему, що значно підвищило ефективність роботи автошколи.

Доцільним є подальше вдосконалення вебзастосунку DreamDrive шляхом впровадження Amazon S3 для більш ефективного зберігання зображень і медіафайлів. Також перспективним напрямком розвитку є розширення функціоналу за рахунок додавання модуля автоматизації документообігу, що дозволить покращити адміністративні процеси автошколи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Положення про автошколу [Електронний ресурс]. – Режим доступу: https://hsc.gov.ua (дата звернення: 12.04.2025).

2) Як отримати посвідчення водія вперше [Електронний ресурс]. – Режим доступу: https://hsc.gov.ua/prava (дата звернення: 12.04.2025).

3) Огляд EdTech-індустрії України 2023 / EdTech Hub [Електронний ресурс]. – Режим доступу: https://edtechhub.org (дата звернення: 14.04.2025).

4) Microsoft Excel [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/en-us/microsoft-365/excel (дата звернення: 16.04.2025).

5) Автошкола «Равлик ПДР Онлайн» [Електронний ресурс]. – Режим доступу: https://pdr-online.com.ua (дата звернення: 20.04.2025).

6) Навчальна платформа «GreenWay» [Електронний ресурс]. – Режим доступу: https://green-way.com.ua (дата звернення: 20.04.2025).

7) CRM для автошколи, програма для розкладу – EasyWeek [Електронний ресурс]. – Режим доступу: https://easyweek.com.ua/solutions/driving-school (дата звернення: 20.04.2025).

8) Онлайн автошкола «Час» [Електронний ресурс]. – Режим доступу: https://autoschool-online.com.ua/ (дата звернення: 21.04.2025).

9) Python [Електронний ресурс]. – Режим доступу: https://www.python.org/ (дата звернення: 26.04.2025).

10) PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/ (дата звернення: 26.04.2025).

11) Django [Електронний ресурс]. – Режим доступу: https://www.djangoproject.com/ (дата звернення: 26.04.2025).

12) Django REST Framework [Електронний ресурс]. – Режим доступу: https://www.django-rest-framework.org/ (дата звернення: 26.04.2025).

13) RESTful API [Електронний ресурс]. – Режим доступу: https://restfulapi.net/ (дата звернення: 26.04.2025).

14) FastAPI [Електронний ресурс]. – Режим доступу: https://fastapi.tiangolo.com/ (дата звернення: 27.04.2025).

15) Docker [Електронний ресурс]. – Режим доступу: https://www.docker.com/ (дата звернення: 28.04.2025).

16) JWT (JSON Web Tokens) [Електронний ресурс]. – Режим доступу: https://jwt.io/ (дата звернення: 30.04.2025).

17) React [Електронний ресурс]. – Режим доступу: https://reactjs.org/ (дата звернення: 30.04.2025).

18) Axios [Електронний ресурс]. – Режим доступу: https://axios-http.com/ (дата звернення: 05.05.2025).

19) Tailwind CSS [Електронний ресурс]. – Режим доступу: https://tailwindcss.com/ (дата звернення: 05.05.2025).

20) Radix UI [Електронний ресурс]. – Режим доступу: https://www.radix-ui.com/ (дата звернення: 05.05.2025).

21) JavaScript [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 05.05.2025).

22) TypeScript [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/ (дата звернення: 05.05.2025).

23) Railway [Електронний ресурс]. – Режим доступу: https://railway.app/ (дата звернення: 05.05.2025).

24) BPMN (Business Process Model and Notation) [Електронний ресурс]. – Режим доступу: https://www.omg.org/spec/BPMN/ (дата звернення: 08.05.2025).

25) LiqPay [Електронний ресурс]. – Режим доступу: https://www.liqpay.ua/ (дата звернення: 08.05.2025).

26) SSL (Secure Sockets Layer) [Електронний ресурс]. – Режим доступу: https://letsencrypt.org/ (дата звернення: 11.05.2025).

27) XSS (Cross-Site Scripting) [Електронний ресурс]. – Режим доступу: https://owasp.org/www-community/attacks/xss/ (дата звернення: 11.05.2025).

28) CSRF (Cross-Site Request Forgery) [Електронний ресурс]. – Режим доступу: https://owasp.org/www-community/attacks/csrf/ (дата звернення: 11.05.2025).

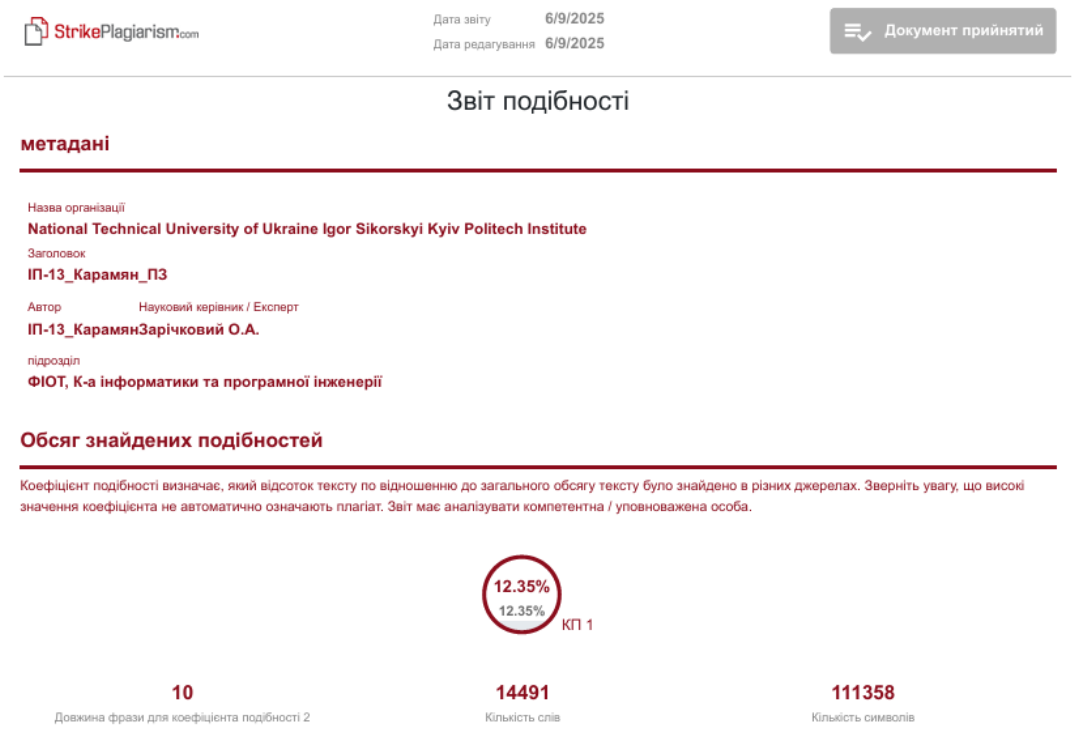
29) OAuth 2.0 [Електронний ресурс]. – Режим доступу: https://oauth.net/2/ (дата звернення: 22.05.2025).

30) Amazon S3 (Simple Storage Service) [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/s3/ (дата звернення: 23.05.2025).

31) SonarCloud [Електронний ресурс]. – Режим доступу: https://sonarcloud.io/ (дата звернення: 29.05.2025).

32) Vercel [Електронний ресурс]. – Режим доступу: https://vercel.com/ (дата звернення: 29.05.2025).

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



StrikePlagiarism.com

Дата звіту 6/9/2025
Дата редагування 6/9/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-13_Карамян_ПЗ
Автор Науковий керівник / Експерт
ІП-13_КарамянЗарічковий О.А.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

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.

12.35%
12.35%
КП 1

10	14491	111358
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ АВТОШКОЛИ

Текст програми

КПІ.ІП-1313.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр ЗАРІЧКОВИЙ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Вартан КАРАМЯН

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду
https://github.com/Vartan14/DreamDrive/tree/main

Реалізація функціональної задачі «Головна сторінка»

Файл backend/public_service/app/core/serializers.py

```
class FilialSerializer(serializers.ModelSerializer):
    class Meta:
        model = Filial
        fields = '__all__'


class ReviewSerializer(serializers.ModelSerializer):
    class Meta:
        model = Review
        fields = '__all__'


class FAQSerializer(serializers.ModelSerializer):
    class Meta:
        model = FAQ
        fields = '__all__'


class LandingElementSerializer(serializers.ModelSerializer):
    class Meta:
        model = LandingElement
        fields = '__all__'


class PricePlanSerializer(serializers.ModelSerializer):
    class Meta:
        model = PricePlan
        fields = '__all__'
```

Файл backend/public_service/app/core/views.py

```
class FilialViewSet(viewsets.ModelViewSet):
    queryset = Filial.objects.all()
    serializer_class = FilialSerializer
    permission_classes = [DefaultRolePermission]


class ReviewViewSet(viewsets.ModelViewSet):
    queryset = Review.objects.all()
    serializer_class = ReviewSerializer
    permission_classes = [DefaultRolePermission]


class FAQViewSet(viewsets.ModelViewSet):
    queryset = FAQ.objects.all()
    serializer_class = FAQSerializer
    permission_classes = [DefaultRolePermission]
```

```
class LandingElementViewSet(viewsets.ModelViewSet):
    queryset = LandingElement.objects.all()
    serializer_class = LandingElementSerializer
    permission_classes = [DefaultRolePermission]


class PricePlanViewSet(viewsets.ModelViewSet):
    queryset = PricePlan.objects.all()
    serializer_class = PricePlanSerializer
    permission_classes = [DefaultRolePermission]
```

Файл backend/public_service/app/core/urls.py

```
router = DefaultRouter()
router.register(r'filials', FilialViewSet)
router.register(r'reviews', ReviewViewSet)
router.register(r'faq', FAQViewSet)
router.register(r'landing-elements', LandingElementViewSet)
router.register(r'price-plans', PricePlanViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

Реалізація функціональної задачі «Система авторизації»

Файл backend/auth_user_service/app/app/urls.py

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path("login/", LoginPage.as_view(), name="login"),
    path('schema/', SpectacularAPIView.as_view(), name='schema'),
    path('docs/', SpectacularSwaggerView.as_view(url_name='schema'), name='docs'),

    path('api/v1/auth/', include("dj_rest_auth.urls")),
    path('api/v1/auth/registration/', include('dj_rest_auth.registration.urls')),

    path('api/v1/auth/password/reset/', dj_rest_auth_views.PasswordResetView.as_view(), name='password_reset'),

    path('api/v1/auth/password/reset/confirm/', dj_rest_auth_views.PasswordResetConfirmView.as_view(),
         name='password_reset_confirm'),
    path('accounts/', include('allauth.urls')),

    path("api/v1/auth/google/", GoogleLogin.as_view(), name="google_login"),
    path("api/v1/auth/google/callback/",GoogleLoginCallback.as_view(), name="google_login_callback"),

    path('api/v1/users/', include('user.urls')),
    path('api/v1/groups/', include('group.urls')),
    path('api/v1/profile/', include('user_profile.urls')),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Файл backend/auth_user_service/app/user/serializers.py

```
User = get_user_model()
```

```
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = get_user_model()
        fields = ('id', 'email', 'password', 'first_name', 'last_name', 'is_paid', 'role',
                  'phone', 'about_me')
        read_only_fields = ('is_paid', 'role')
        extra_kwargs = {
            'password': {'write_only': True, 'min_length': 8},
        }

    def create(self, validated_data):
        return get_user_model().objects.create_user(**validated_data)

    def update(self, instance, validated_data):
        if 'email' in validated_data and validated_data['email'] != instance.email:
            raise serializers.ValidationError({
                'email': 'Email address cannot be changed.'
            })
        password = validated_data.pop('password', None)
        user = super().update(instance, validated_data)

        if password:
            user.set_password(password)
            user.save()

        return user


class AdminUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = get_user_model()
        fields = '__all__'
        extra_kwargs = {
            'password': {'write_only': True, 'min_length': 8},
        }

    def create(self, validated_data):
        return get_user_model().objects.create_user(**validated_data)

    def update(self, instance, validated_data):
        if 'email' in validated_data and validated_data['email'] != instance.email:
```

```
            raise serializers.ValidationError({
                'email': 'Email address cannot be changed.'
            })
        password = validated_data.pop('password', None)
        user = super().update(instance, validated_data)

        if password:
            user.set_password(password)
            user.save()
        return user

class MyTokenObtainPairSerializer(TokenObtainPairSerializer):
    @classmethod
    def get_token(cls, user):
        token = super().get_token(user)
        token['email'] = user.email
        token['role'] = user.role
        return token

class MyRegisterSerializer(RegisterSerializer):
    first_name = serializers.CharField(required=True)
    last_name = serializers.CharField(required=True)

    def validate_email(self, email):
        if get_user_model().objects.filter(email=email).exists():
            raise serializers.ValidationError("A user with this email already exists.")
        return email

    def custom_signup(self, request, user):
        user.first_name = self.validated_data.get('first_name', '')
        user.last_name = self.validated_data.get('last_name', '')
        user.save()

class CustomPasswordResetSerializer(PasswordResetSerializer):
    @property
    def password_reset_form_class(self):
        return CustomAllAuthPasswordResetForm

class StudentPaymentUpdateSerializer(serializers.Serializer):
    user_id = serializers.IntegerField()

    def validate_user_id(self, value):
```

```
        if not User.objects.filter(id=value).exists():
            raise serializers.ValidationError("User not found")
        return value

    def update(self, instance, validated_data):
        pass

    def create(self, validated_data):
        user = User.objects.get(id=validated_data['user_id'])
        user.is_paid = True
        user.save()
        StudentProfile.objects.get_or_create(user=user)
        return user
```

Файл backend/auth_user_service/app/user/views.py

```
class UserCreateView(generics.CreateAPIView):
    serializer_class = UserSerializer
    permission_classes = []

class ManageUserView(generics.RetrieveUpdateAPIView):
    serializer_class = UserSerializer
    authentication_classes = [authentication.JWTAuthentication]
    permission_classes = [permissions.IsAuthenticated]

    def get_object(self):
        return self.request.user

    def get(self, request):
        serializer = UserWithProfileSerializer(request.user)
        return Response(serializer.data)

class UserAdminViewSet(viewsets.ModelViewSet):
    queryset = get_user_model().objects.all()
    serializer_class = AdminUserSerializer
    permission_classes = [permissions.IsAdminUser]

class UpdateStudentPaymentStatusView(views.APIView):
    authentication_classes = []
    permission_classes = [IsPaymentMicroservice]

    def post(self, request):
        serializer = StudentPaymentUpdateSerializer(data=request.data)
```

```
        if serializer.is_valid():
            user = serializer.save()
            return Response({
                "message": f"User {user.id} payment status updated successfully."
            })
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

Файл backend/auth_user_service/app/user/views.py

```
class TeacherProfile(models.Model):
    class TeachingType(models.TextChoices):
        THEORY = 'theory', 'Theory'
        PRACTICE = 'practice', 'Practice'
        BOTH = 'theory_practice', 'Theory and Practice'

    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='teacher_profile')
    type = models.CharField(max_length=20, choices=TeachingType.choices)

    def clean(self):
        if self.user.role != User.Role.TEACHER:
            raise ValidationError("TeacherProfile can be assigned only with the role 'teacher'.")

    def save(self, *args, **kwargs):
        self.clean()
        super().save(*args, **kwargs)

    def __str__(self):
        return f"{self.user.get_full_name()} ({self.get_type_display()})"

    def get_email(self):
        return self.user.email if self.user else None

    def get_full_name(self):
        return self.user.get_full_name() if self.user else None

class Group(models.Model):
    class GroupType(models.TextChoices):
        THEORY = 'theory', 'Theory'
        PRACTICE = 'practice', 'Practice'
        THEORY_PRACTICE = 'theory_practice', 'Theory and Practice'

    name = models.CharField(max_length=100)
    description = models.TextField(blank=True, default='')
```

```
    driving_category = models.CharField(max_length=5)
    teacher = models.ForeignKey(TeacherProfile, on_delete=models.DO_NOTHING, related_name='groups',
null=True, blank=True)
    filial_id = models.PositiveIntegerField(null=True, blank=True)
    type = models.CharField(max_length=20, choices=GroupType.choices)

    def __str__(self):
        return f"{self.name} - {self.get_type_display()}"

class StudentProfile(models.Model):
    class StudentType(models.TextChoices):
        THEORY = 'theory', 'Theory'
        PRACTICE = 'practice', 'Practice'

    user = models.OneToOneField(User, on_delete=models.DO_NOTHING, related_name='student_profile')
    group = models.ForeignKey(Group, on_delete=models.DO_NOTHING, null=True, blank=True,
related_name='students')
    type = models.CharField(max_length=20, choices=StudentType.choices, default=StudentType.THEORY)
    progress = models.IntegerField(default=0, blank=True, null=True, validators=[MinValueValidator(0),
MaxValueValidator(100)])

    def clean(self):
        if self.user.role != User.Role.STUDENT:
            raise ValidationError("StudentProfile can be assigned only with the role 'student'.")

    def save(self, *args, **kwargs):
        self.clean()
        super().save(*args, **kwargs)

    def __str__(self):
        return self.user.get_full_name()

    def get_email(self):
        return self.user.email if self.user else None

    def get_full_name(self):
        return self.user.get_full_name() if self.user else None
```

Файл backend/auth_user_service/app/user_profile/serializers.py

```
class StudentProfileSerializer(serializers.ModelSerializer):
    first_name = serializers.CharField(source='user.first_name', read_only=True)
    last_name = serializers.CharField(source='user.last_name', read_only=True)
```

```
    email = serializers.EmailField(source='user.email', read_only=True)

    class Meta:
        model = StudentProfile
        fields = ['id', 'user_id', 'first_name', 'last_name', 'email', 'progress', 'type', 'group_id']

    def validate_user(self, value):
        if value.role != value.Role.STUDENT:
            raise serializers.ValidationError("User must have the role 'student' to create a StudentProfile.")
        return value

class TeacherProfileSerializer(serializers.ModelSerializer):
    id = serializers.IntegerField(source='user.id', read_only=True)
    teacher_id = serializers.IntegerField(source='id', read_only=True)
    groups = serializers.SerializerMethodField()
    name = serializers.CharField(source='get_full_name', read_only=True)

    class Meta:
        model = TeacherProfile
        fields = ('teacher_id', 'id', 'name', 'type') + ('groups',)

    def get_groups(self, obj):
        groups = Group.objects.filter(teacher=obj)
        return [{'id': group.id, 'name': group.name, 'type': group.type} for group in groups]

    def validate_user(self, value):
        if value.role != value.Role.TEACHER:
            raise serializers.ValidationError("User must have the role 'teacher' to create a TeacherProfile.")
        return value

class UserWithProfileSerializer(UserSerializer):
    profile = serializers.SerializerMethodField()

    class Meta(UserSerializer.Meta):
        fields = UserSerializer.Meta.fields + ('profile',)

    def get_profile(self, user):
        if user.role == User.Role.STUDENT and hasattr(user, 'student_profile'):
            return StudentProfileSerializer(user.student_profile).data
        elif user.role == User.Role.TEACHER and hasattr(user, 'teacher_profile'):
            return TeacherProfileSerializer(user.teacher_profile).data
        return None
```

Файл backend/auth_user_service/app/user_profile/views.py

```
class StudentProfileViewSet(viewsets.ModelViewSet):
    queryset = StudentProfile.objects.all()
    serializer_class = StudentProfileSerializer
    permission_classes = [RolePermission.allow_roles('admin', 'teacher')]

class TeacherProfileViewSet(viewsets.ModelViewSet):
    queryset = TeacherProfile.objects.all()
    serializer_class = TeacherProfileSerializer
    permission_classes = [DefaultRolePermission]

    def get_object(self):
        user_id = self.kwargs.get('pk')
        return TeacherProfile.objects.get(user__id=user_id)
```

Файл

Реалізація функціональної задачі «Платежі»

Файл backend/payment_service/app/payments/serializers.py

```
class CreatePaymentSerializer(serializers.Serializer):
    amount = serializers.DecimalField(max_digits=10, decimal_places=2)
    description = serializers.CharField(required=False, default="Оплата в DreamDrive")

class LiqPayCallbackSerializer(serializers.Serializer):
    data = serializers.CharField()
    signature = serializers.CharField()

class PaymentStatusSerializer(serializers.ModelSerializer):
    class Meta:
        model = Payment
        fields = ["liqpay_order_id", "amount", "status", "created_at", "description"]

class PaymentHistorySerializer(serializers.ModelSerializer):
    class Meta:
        model = Payment
        fields = ["liqpay_order_id", "amount", "status", "created_at", "description"]
        read_only_fields = ["liqpay_order_id", "amount", "status", "created_at", "description"]
```

Файл backend/payment_service/app/payments/views.py

```
class CreatePaymentView(APIView):
    permission_classes = [RolePermission.allow_roles('student')]
    serializer_class = CreatePaymentSerializer

    def post(self, request):
        user = request.user
        amount = request.data.get("amount")
        description = request.data.get("description", "Оплата в DreamDrive")
        order_id = str(uuid.uuid4())

        if not amount:
            return Response({"error": "Amount is required"}, status=status.HTTP_400_BAD_REQUEST)

        data = {
            "public_key": settings.LIQPAY_PUBLIC_KEY,
            "version": "3",
            "action": "pay",
            "amount": amount,
            "currency": "UAH",
            "description": description,
            "order_id": order_id,
            "sandbox": 1,
            "result_url": settings.LIQPAY_RESULT_URL,
            "server_url": settings.LIQPAY_CALLBACK_URL,
        }

        encoded_data, signature = generate_signature(data)

        Payment.objects.create(
            user_id=user.id,
            amount=amount,
            description=description,
            liqpay_order_id=order_id
        )

        return Response({
            "data": encoded_data,
            "signature": signature,
            "liqpay_url": "https://www.liqpay.ua/api/3/checkout",
            "user_id": user.id
        })
```

```
class LiqPayCallbackView(APIView):
    permission_classes = [AllowAny]
    serializer_class = LiqPayCallbackSerializer

    def post(self, request):
        data = request.data.get("data")
        signature = request.data.get("signature")

        if not data or not signature:
            return Response({'error': 'Missing data or signature'}, status=status.HTTP_400_BAD_REQUEST)

        expected_signature = base64.b64encode(
            hashlib.sha1((settings.LIQPAY_PRIVATE_KEY + data +
settings.LIQPAY_PRIVATE_KEY).encode()).digest()
        ).decode()

        if signature != expected_signature:
            return Response({'error': 'Invalid signature'}, status=status.HTTP_400_BAD_REQUEST)

        try:
            decoded = json.loads(base64.b64decode(data).decode())
        except Exception:
            return Response({'error': 'Failed to decode data'}, status=status.HTTP_400_BAD_REQUEST)

        order_id = decoded.get("order_id")
        if not order_id:
            return Response({'error': 'Missing order_id'}, status=status.HTTP_400_BAD_REQUEST)

        try:
            payment = Payment.objects.get(liqpay_order_id=order_id)
            payment_status = decoded.get("status")
            if payment_status in ('success', 'sandbox'):
                payment.status = 'success'
                try:
                    response = requests.post(
                        url=settings.AUTH_SERVICE_URL,
```

Файл backend/payment_service/app/payments/urls.py

```
urlpatterns = [
    path('create-payment/', views.CreatePaymentView.as_view(), name='create_payment'),
    path('callback/', views.LiqPayCallbackView.as_view(), name='liqpay_callback'),
```

```
    path('status/', views.PaymentStatusView.as_view(), name='payment-status'),
    path('payment-history/', views.PaymentHistoryView.as_view(), name='payment-history'),
]
```

Реалізація функціональної задачі «Навчальні матеріали»

Файл backend/learning_service/app/pdr/serializers.py

```
from rest_framework import serializers
from .models import TrafficRule, RuleSection, RoadVisualElement, RoadVisualGroup

class RoadVisualGroupSerializer(serializers.ModelSerializer):
    class Meta:
        model = RoadVisualGroup
        fields = ['id', 'type', 'number', 'title']

class RoadVisualElementSerializer(serializers.ModelSerializer):
    group_number = serializers.CharField(source='group.number', read_only=True)

    class Meta:
        model = RoadVisualElement
        fields = ['id', 'element_id', 'name', 'text', 'image', 'group_number']

class RuleSectionSerializer(serializers.ModelSerializer):
    class Meta:
        model = RuleSection
        fields = ['id', 'number', 'title']

class TrafficRuleSerializer(serializers.ModelSerializer):
    section_number = serializers.CharField(source='section.number', read_only=True)

    class Meta:
        model = TrafficRule
        fields = ['id','rule_id', 'text', 'section_number']
```

Файл backend/learning_service/app/pdr/views.py

```
from drf_spectacular.utils import extend_schema
from rest_framework import viewsets

from core.permissions import RolePermission, DefaultRolePermission
from .models import TrafficRule, RuleSection, RoadVisualElement, RoadVisualGroup
from .serializers import (
```

```
    TrafficRuleSerializer,
    RuleSectionSerializer,
    RoadVisualElementSerializer,
    RoadVisualGroupSerializer,
)

@extend_schema(tags=["PDR / Sing And Marking Groups"])
class RoadVisualGroupViewSet(viewsets.ModelViewSet):
    queryset = RoadVisualGroup.objects.all()
    serializer_class = RoadVisualGroupSerializer
    permission_classes = [DefaultRolePermission]

@extend_schema(tags=["PDR / Signs and Markings"])
class RoadVisualElementViewSet(viewsets.ModelViewSet):
    queryset = RoadVisualElement.objects.all()
    serializer_class = RoadVisualElementSerializer
    permission_classes = [DefaultRolePermission]

    def get_queryset(self):
        queryset = RoadVisualElement.objects.all()
        group_id = self.request.query_params.get('group')
        if group_id:
            try:
                group_id = int(group_id)
            except ValueError:
                raise ValueError("Group id must be an integer.")
            queryset = queryset.filter(group__id=group_id)
        return queryset

@extend_schema(tags=["PDR / Rule Sections"])
class RuleSectionViewSet(viewsets.ModelViewSet):
    queryset = RuleSection.objects.all()
    serializer_class = RuleSectionSerializer
    permission_classes = [DefaultRolePermission]

@extend_schema(tags=["PDR / Traffic Rules"])
class TrafficRuleViewSet(viewsets.ModelViewSet):
    queryset = TrafficRule.objects.all()
    serializer_class = TrafficRuleSerializer
    permission_classes = [DefaultRolePermission]

    def get_queryset(self):
```

```
        queryset = TrafficRule.objects.all()
        section_number = self.request.query_params.get('section')
        if section_number:
            try:
                section_number = int(section_number)
            except ValueError:
                raise ValueError("Section number must be an integer.")
            queryset = queryset.filter(section__number=section_number)
        return queryset
```

Файл backend/learning_service/app/pdr/urls.py

```
router = DefaultRouter()
router.register(r'rules', TrafficRuleViewSet)
router.register(r'rule-sections', RuleSectionViewSet)
router.register(r'signs-and-markings', RoadVisualElementViewSet)
router.register(r'sign-and-marking-groups', RoadVisualGroupViewSet)

urlpatterns = router.urls
```

Реалізація функціональної задачі «Розклад занять»

Файл backend/schedule_service/app/schedule/serializers.py

```
from rest_framework import serializers
from .models import TheoryLesson, PracticeLesson


class TheoryLessonSerializer(serializers.ModelSerializer):
    end_time = serializers.SerializerMethodField()

    class Meta:
        model = TheoryLesson
        fields = [
            'id', 'title', 'start_time', 'end_time', 'duration',
            'filial_id', 'instructor_id', 'instructor_name',
            'group_id', 'is_online', 'status'
        ]
        read_only_fields = ('instructor_id', 'created_at')

    def get_end_time(self, obj):
```

```
        return obj.start_time + obj.duration if obj.start_time and obj.duration else None


class PracticeLessonSerializer(serializers.ModelSerializer):
    end_time = serializers.SerializerMethodField()

    class Meta:
        model = PracticeLesson
        fields = [
            'id', 'title', 'start_time', 'end_time', 'duration',
            'filial_id', 'instructor_id', 'instructor_name',
            'status', 'location', 'car', 'student_id'
        ]
        read_only_fields = ('instructor_id', 'created_at')

    def get_end_time(self, obj):
        return obj.start_time + obj.duration if obj.start_time and obj.duration else None
```

Файл backend/schedule_service/app/schedule/views.py

```
@extend_schema(tags=["Student"])
class ReadOnlyTheoryLessonViewSet(viewsets.ReadOnlyModelViewSet):
    queryset = TheoryLesson.objects.all()
    serializer_class = TheoryLessonSerializer
    permission_classes = [RolePermission.allow_roles('student')]

    def get_queryset(self):
        group_id = self.request.headers.get('X-User-Group-Id')
        if not group_id:
            raise ValueError("Header 'X-User-Group-Id' is required.")
        return TheoryLesson.objects.filter(group_id=group_id)


@extend_schema(tags=["Student"])
class ReadOnlyPracticeLessonViewSet(viewsets.ReadOnlyModelViewSet):
    queryset = PracticeLesson.objects.all()
    serializer_class = PracticeLessonSerializer
    permission_classes = [RolePermission.allow_roles('student')]

    def get_queryset(self):
        user_id = self.request.user.id
        return PracticeLesson.objects.filter(student_id=user_id)
```

```
@extend_schema(tags=["Student"])
class ListAvailablePracticeLessons(generics.ListAPIView):
    queryset = PracticeLesson.objects.filter(status='available')
    serializer_class = PracticeLessonSerializer
    permission_classes = [RolePermission.allow_roles('student')]


@extend_schema(tags=["Student"], methods=["PATCH"])
class BookPracticeLessonView(generics.UpdateAPIView):
    queryset = PracticeLesson.objects.all()
    serializer_class = PracticeLessonSerializer
    permission_classes = [RolePermission.allow_roles('student')]

    def perform_update(self, serializer):
        if serializer.instance.status != 'available':
            raise ValidationError('This lesson is not available now')
        if serializer.instance.student_id is not None:
            raise ValidationError('This lesson is already booked by another student')
        serializer.save(status='booked', student_id=self.request.user.id)

    def put(self, request, *args, **kwargs):
        return Response({"detail": "Method PUT not allowed."},
status=status.HTTP_405_METHOD_NOT_ALLOWED)


@extend_schema(tags=["Teacher"])
class TheoryLessonViewSet(viewsets.ModelViewSet):
    queryset = TheoryLesson.objects.none()
    serializer_class = TheoryLessonSerializer
    permission_classes = [RolePermission.allow_roles('teacher', 'admin')]

    def get_queryset(self):
        if self.request.user.role != 'admin':
            instructor_id = self.request.user.id
            return TheoryLesson.objects.filter(instructor_id=instructor_id)
        else:
            return TheoryLesson.objects.all()

    def perform_create(self, serializer):
        instructor_id = self.request.query_params.get('instructor_id')
        if not instructor_id:
```

```
            serializer.save(instructor_id=self.request.user.id)
        else:
            serializer.save(instructor_id=instructor_id)


@extend_schema(tags=["Teacher"])
class PracticeLessonViewSet(viewsets.ModelViewSet):
    queryset = PracticeLesson.objects.none()
    serializer_class = PracticeLessonSerializer
    permission_classes = [RolePermission.allow_roles('teacher', 'admin')]

    def get_queryset(self):
        if self.request.user.role != 'admin':
            instructor_id = self.request.user.id
            return PracticeLesson.objects.filter(instructor_id=instructor_id)
        else:
            return PracticeLesson.objects.all()

    def perform_create(self, serializer):
        instructor_id = self.request.query_params.get('instructor_id')
        if not instructor_id:
            serializer.save(instructor_id=self.request.user.id)
        else:
            serializer.save(instructor_id=instructor_id)


@extend_schema(tags=["Teacher"])
class InstructorCalendarEvents(generics.ListAPIView):
    permission_classes = [RolePermission.allow_roles('teacher')]
    serializer_class = LessonCalendarSerializer

    def get_queryset(self):
        date = self.request.query_params.get('date')
        if not date:
            raise ValidationError("Date parameter is required.")
        start = f"{date}T00:00:00"
        end = f"{date}T23:59:59"
        try:
            instructor_id = self.request.user.id
            theory = TheoryLesson.objects.filter(instructor_id=instructor_id, start_time__range=[start, end])
            practice = PracticeLesson.objects.filter(instructor_id=instructor_id, start_time__range=[start, end])
        except ValueError:
```

```
            raise ValidationError("Invalid date format. Expected format is yyyy-mm-dd.")
        return list(theory) + list(practice)


@extend_schema(tags=["Teacher"])
class InstructorTimelineView(generics.ListAPIView):
    permission_classes = [RolePermission.allow_roles('teacher')]
    serializer_class = LessonTimelineSerializer

    def get_queryset(self):
        instructor_id = self.request.user.id
        return sorted(
            list(TheoryLesson.objects.filter(instructor_id=instructor_id)) +
            list(PracticeLesson.objects.filter(instructor_id=instructor_id)),
            key=lambda x: x.start_time
        )

@extend_schema(tags=["Admin"])
class AdminScheduleView(generics.ListAPIView):
    permission_classes = [RolePermission.allow_roles('admin')]
    serializer_class = LessonTimelineSerializer

    def get_queryset(self):
        instructor_id = self.request.query_params.get('instructor_id')
        if not instructor_id:
            raise ValueError("Instructor parameter is required.")
        return sorted(
            list(TheoryLesson.objects.filter(instructor_id=instructor_id)) +
            list(PracticeLesson.objects.filter(instructor_id=instructor_id)),
            key=lambda x: x.start_time
        )
```

Файл backend/schedule_service/app/schedule/views.py

```
theory_router = DefaultRouter()
practice_router = DefaultRouter()
theory_router.register(r'theory-lessons', TheoryLessonViewSet, basename='theory')
practice_router.register(r'practice-lessons', PracticeLessonViewSet, basename='practice')

urlpatterns = [
    path('student/theory-lessons/<int:pk>', ReadOnlyTheoryLessonViewSet.as_view({"get": "retrieve"})),
    path('student/theory-lessons/', ReadOnlyTheoryLessonViewSet.as_view({"get": "list"})),
```

```
    path('student/practice-lessons/<int:pk>', ReadOnlyPracticeLessonViewSet.as_view({"get": "retrieve"})),
    path('student/practice-lessons/', ReadOnlyPracticeLessonViewSet.as_view({"get": "list"})),

    path('student/available-practice-lessons/', ListAvailablePracticeLessons.as_view()),
    path('student/book-practice-lesson/<int:pk>/', BookPracticeLessonView.as_view()),

    path('teacher/', include(theory_router.urls)),
    path('teacher/', include(practice_router.urls)),

    path('teacher/calendar/', InstructorCalendarEvents.as_view()),
    path('teacher/lesson-timeline/', InstructorTimelineView.as_view()),
    path('admin/calendar/', AdminScheduleView.as_view()),
]
```

Реалізація функціональної задачі «Тестування»

Файл backend/learning_service/app/tickets/serializers.py

```
class AnswerSerializer(serializers.ModelSerializer):
    class Meta:
        model = Answer
        fields = ['id', 'text', 'is_correct']

class QuestionSerializer(serializers.ModelSerializer):
    section_number = serializers.IntegerField(source='section.number', read_only=True)
    section_title = serializers.CharField(source='section.title', read_only=True)
    answers = AnswerSerializer(many=True, read_only=True)

    class Meta:
        model = Question
        fields = ['id', 'section_number', 'section_title', 'ticket_number', 'question_number',
                  'text', 'reply_text', 'rule', 'image', 'answers']

class TicketSerializer(serializers.ModelSerializer):
    class Meta:
        model = Ticket
        fields = ['id', 'name', 'ticket_number', 'is_custom', 'created_at']

class TicketQuestionSerializer(serializers.ModelSerializer):
    class Meta:
        model = TicketQuestion
        fields = ['id', 'ticket', 'question', 'order']
```

Файл backend/learning_service/app/tickets/views.py

```
class QuestionPagination(PageNumberPagination):
    page_size = 20

class QuestionViewSet(viewsets.ModelViewSet):
    queryset = Question.objects.all()
    serializer_class = QuestionSerializer
    permission_classes = [DefaultRolePermission]
    pagination_class = QuestionPagination

    def get_queryset(self):
        queryset = Question.objects.all()
        section = self.request.query_params.get('section')
        if section is None:
            section = '1'
        queryset = queryset.filter(section__number=section)
        return queryset

class AnswerViewSet(viewsets.ModelViewSet):
    queryset = Answer.objects.all()
    serializer_class = AnswerSerializer
    permission_classes = [RolePermission.allow_roles('admin', 'teacher')]

class TicketViewSet(viewsets.ModelViewSet):
    queryset = Ticket.objects.all()
    serializer_class = TicketSerializer
    permission_classes = [RolePermission.allow_roles('admin', 'teacher')]

class TicketQuestionViewSet(viewsets.ModelViewSet):
    queryset = TicketQuestion.objects.all()
    serializer_class = TicketQuestionSerializer
    permission_classes = [RolePermission.allow_roles('admin', 'teacher')]
```

Файл backend/learning_service/app/tickets/urls.py

```
router = DefaultRouter()

router.register(r'', TicketViewSet)
router.register(r'questions', QuestionViewSet)
router.register(r'answers', AnswerViewSet)
router.register(r'ticket-questions', TicketQuestionViewSet)
```

```
urlpatterns = router.urls
```

Файл backend/learning_service/app/testing/serializers.py

```
class TicketTestSessionSerializer(serializers.ModelSerializer):
    ticket_id = serializers.PrimaryKeyRelatedField(
        queryset=Ticket.objects.all(),
        source='ticket',
        write_only=True
    )
    ticket_number = serializers.ReadOnlyField(source='ticket.ticket_number')
    started_at = serializers.DateTimeField(read_only=True)
    completed_at = serializers.DateTimeField(read_only=True)

    class Meta:
        model = TicketTestSession
        fields = [
            'id',
            'user_id',
            'ticket_id',
            'ticket_number',
            'started_at',
            'completed_at',
            'is_passed',
            'current_question_index',
            'mistakes_count'
        ]
        read_only_fields = ['id', 'user_id', 'ticket_number', 'started_at', 'completed_at', 'is_passed',
'current_question_index', 'mistakes_count']

    def create(self, validated_data):
        user_id = self.context['user_id']
        return TicketTestSession.objects.create(user_id=user_id, **validated_data)
```

Файл backend/learning_service/app/testing/views.py

```
class StartTicketTestView(APIView):
    permission_classes = [IsAuthenticated]

    def post(self, request):
        ticket_id = request.data.get("ticket_id")
        use_random = request.data.get("random", False)
```

```
if use_random:
    tickets = Ticket.objects.all()
    if not tickets.exists():
        return Response({"detail": "No tickets available."}, status=status.HTTP_404_NOT_FOUND)
    ticket = random.choice(tickets)
else:
    if not ticket_id:
        return Response({"detail": "ticket_id is required unless random=true."},
                        status=status.HTTP_400_BAD_REQUEST)
    ticket = get_object_or_404(Ticket, id=ticket_id)

session = TicketTestSession.objects.create(
    user_id=request.user.id,
    ticket=ticket
)

ticket_questions = ticket.questions.all().order_by("order")
questions_payload = []

for tq in ticket_questions:
    TicketTestAnswer.objects.create(
        session=session,
        question=tq.question
    )

    answers = Answer.objects.filter(question=tq.question).values("id", "text", "is_correct")
    questions_payload.append({
        "id": tq.question.id,
        "question_number": tq.question.question_number,
        "text": tq.question.text,
        "image": tq.question.image.url if tq.question.image else None,
        "answers": list(answers)
    })

return Response({
    "session_id": session.id,
    "ticket_id": ticket.id,
    "ticket_number": ticket.ticket_number,
    "questions": questions_payload
}, status=status.HTTP_201_CREATED)
```

```
class SubmitTicketTestView(APIView):
    permission_classes = [IsAuthenticated]

    def post(self, request):
        session_id = request.data.get("session_id")
        answers_data = request.data.get("answers", [])

        if not session_id or not answers_data:
            return Response(
                {"detail": "session_id and answers are required."},
                status=status.HTTP_400_BAD_REQUEST
            )

        try:
            session = TicketTestSession.objects.get(id=session_id, user_id=request.user.id)
        except TicketTestSession.DoesNotExist:
            return Response({"detail": "Test session not found."}, status=status.HTTP_404_NOT_FOUND)

        if session.completed_at:
            return Response({"detail": "Test already completed."}, status=status.HTTP_400_BAD_REQUEST)

        mistakes = 0
        for answer_entry in answers_data:
            question_id = answer_entry.get("question_id")
            selected_answer_id = answer_entry.get("selected_answer_id")
            answered_at = answer_entry.get("answered_at")

            if not question_id or not selected_answer_id:
                continue

            try:
                answer = Answer.objects.get(id=selected_answer_id, question_id=question_id)
            except Answer.DoesNotExist:
                continue

            test_answer = TicketTestAnswer.objects.get(session=session, question_id=question_id)
            test_answer.selected_answer = answer
            test_answer.is_correct = answer.is_correct
            test_answer.answered_at = answered_at
            test_answer.save()

            if not answer.is_correct:
```

```
            mistakes += 1

        session.mistakes_count = mistakes
        session.completed_at = timezone.now()
        session.is_passed = mistakes <= 2
        session.save()

        return Response({
            "session_id": session.id,
            "completed_at": session.completed_at,
            "mistakes": mistakes,
            "is_passed": session.is_passed,
        }, status=status.HTTP_200_OK)
```

Файл backend/learning_service/app/testing/urls.py

```
urlpatterns = [
    path('start/', StartTicketTestView.as_view(), name='start_ticket_test'),
    path('submit/', SubmitTicketTestView.as_view(), name='submit_ticket_test'),
]
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ АВТОШКОЛИ

Програма та методика тестування

КПІ.ІП-1313.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр ЗАРІЧКОВИЙ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Вартан КАРАМЯН

Київ – 2025

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є Об’єктом випробування є вебзастосунок DreamDrive, призначений для організації та управління навчальним процесом в автошколі. Застосунок реалізовано за мікросервісною архітектурою та складається з клієнтської частини та серверної частини, які взаємодіють через API Gateway.

Клієнтська частина розроблена мовою програмування TypeScript з використанням фреймворку React. Серверна частина побудована на базі Python з використанням фреймворку Django. Застосунок орієнтований на роботу у сучасних браузерах, зокрема Google Chrome, Mozilla Firefox, на операційних системах Windows, macOS і Linux.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка валідації введених користувачами даних;
- перевірка збереження та обробки даних у базах даних кожного мікросервісу;
- перевірка сумісності вебзастосунку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox);
- перевірка сумісності застосунку з різними операційними системами (Windows, Linux, MacOS);
- виявлення помилок, збоїв і логічних помилок з метою їх усунення;
- перевірка зручності та логічності користувацького інтерфейсу;
- оцінка швидкості реакції системи на користувацькі дії та навантаження.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення DreamDrive використовуються такі методи:

- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;
- тестування «чорної скриньки» – об’єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;
- тестування «сірої скриньки» – об’єктом тестування тут є деякі особливості внутрішньої поведінки програми. Перевіряється коректність вихідних даних при заданих вхідних, застосовується для тестування окремих алгоритмів (функцій).

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Під час проведення тестування будуть використовуватись наступні допоміжні засоби:

- середовище Postman для перевірки REST API;
- розширення браузера DevTools для інспекції UI та мережевих запитів.

Порядок проведення тестування буде наступним:

– первинне динамічне тестування функціоналу згідно з технічними вимогами;

– перевірка адаптивності інтерфейсу на пристроях з різними роздільними здатностями екрана;

– тестування ключових бізнес-функцій: реєстрації, авторизації, оплат, перегляду матеріалів, запису на заняття та управління розкладом, проходження тестів;

– тестування коректності обробки помилок і повідомлень для користувача;

– перевірка сумісності у браузерах Chrome, Firefox, Opera;

– оцінка зручності та логіки інтерфейсу користувача з точки зору досвіду користувача (UX).

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ АВТОШКОЛИ

Керівництво користувача

КПІ.ІП-1313.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр ЗАРІЧКОВИЙ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Вартан КАРАМЯН

Київ – 2025

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

DreamDrive — це сучасний вебзастосунок, розроблений для цифровізації освітнього процесу в автошколі. Програма дозволяє ефективно організувати навчання учнів, взаємодію з інструкторами та адміністрування внутрішніх процесів автошколи через консолідацію декількох програмних рішень у єдину платформу.

Застосунок забезпечує наступні основні функції:

– головна сторінка, де розміщено публічну інформацію про автошколу, філії, викладачів, ціни та відгуки;

– реєстрація та авторизація користувачів з різними ролями: учень, інструктор, адміністратор;

– платіжний модуль, який дозволяє здійснювати онлайн-оплату навчання та переглядати платежі;

– навчальний модуль, що включає матеріали з Правил дорожнього руху (ПДР) та інструменти для управління ними;

– модуль розкладу з можливістю перегляду та бронювання занять студентами та інтерактивний календар для викладачів та адміністратора;

– тестування знань за офіційними білетами або індивідуальними добірками питань.

Остаточна збірка розгортається як набір мікросервісів, що взаємодіють через REST API. Кожен мікросервіс відповідає за окрему частину функціоналу. Інсталяційна версія складається з фронтенд-клієнта (React.js) та бекенд-частини на основі Django та FastAPI, які можуть розгортатися через Docker.

DreamDrive є зручним інструментом для всіх учасників освітнього процесу автошколи та підвищує ефективність її роботи.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;

– об‘єм ОЗП: 8 Гб;

– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;

– об‘єм ОЗП: 16 Гб;

– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

2.2 Завантаження застосунку

DreamDrive — це розгорнутий вебзастосунок, доступ до якого здійснюється через веббраузер за посиланням.

2.3 Перевірка коректної роботи

Для перевірки коректної роботи розгорнутого вебзастосунку DreamDrive необхідно перейти за вказаним URL-посиланням і впевнитися, що головна сторінка завантажується без помилок, авторизація користувача працює належним чином, а також доступні основні функції відповідно до ролі (студент, викладач, адміністратор).

3 ВИКОНАННЯ ПРОГРАМИ

У вебзастосунку передбачено три ролі, для кожної з яких буде окремо розглянуто процес виконання програми з усіма доступними функціями. Окремо розглянемо спільний функціонал для всіх ролей.

3.1 Студент

Користувачі можуть знайти посилання на вебзастосунок у пошуковому рядку браузера та потрапити на головну сторінку (рисунок 3.1).

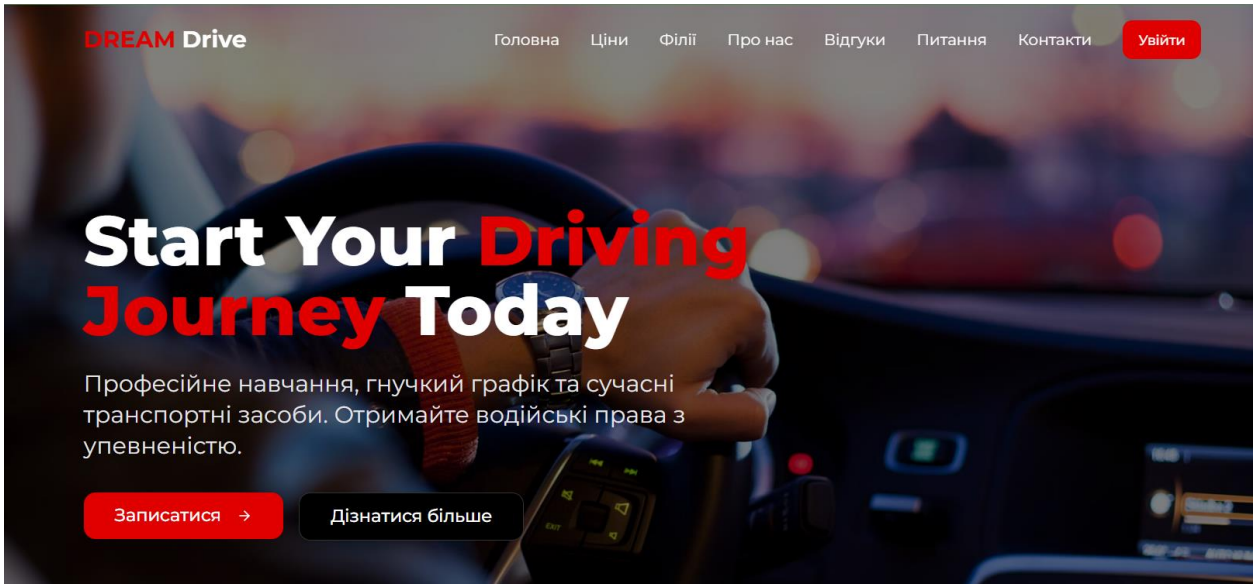


Рисунок 3.1 – Головна сторінка

Потенційний клієнт автошколи переглядає доступні тарифні плани на сторінці з цінами (рисунок 3.2).

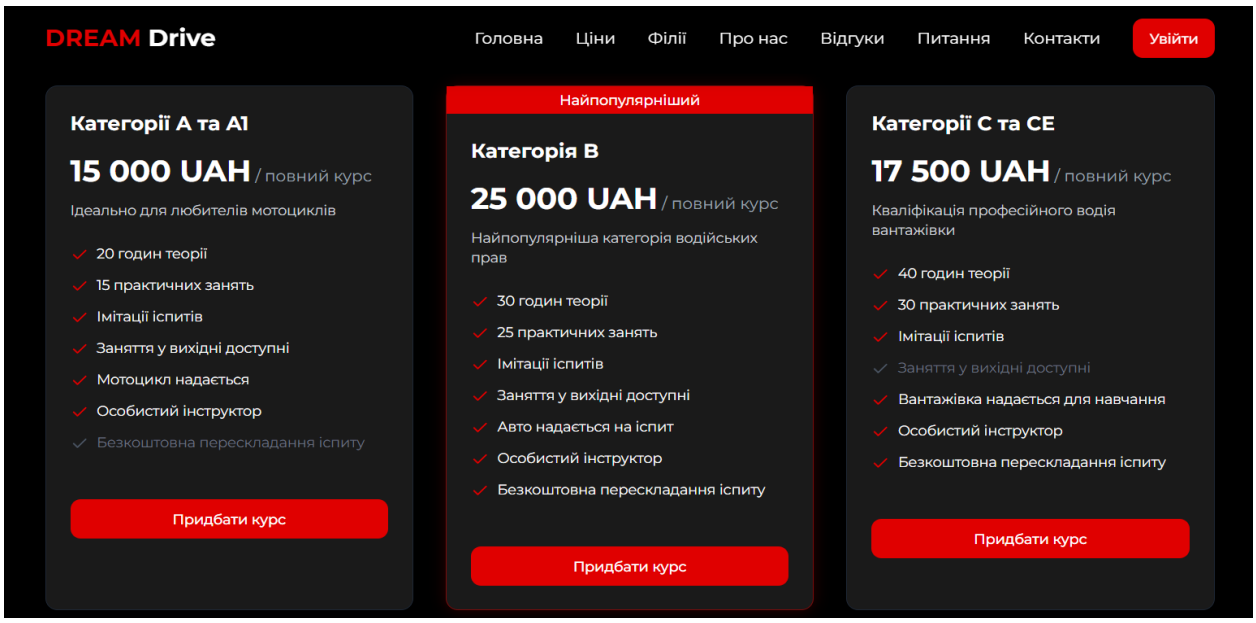


Рисунок 3.2 – Сторінка з цінами

Якщо натиснути «Придбати курс», то потрапляємо на сторінку реєстрації (рисунок 3.3). У відповідні поля вписуємо своє ім’я, прізвище, електронну пошту і пароль двічі та натискаємо «Зареєструватися».

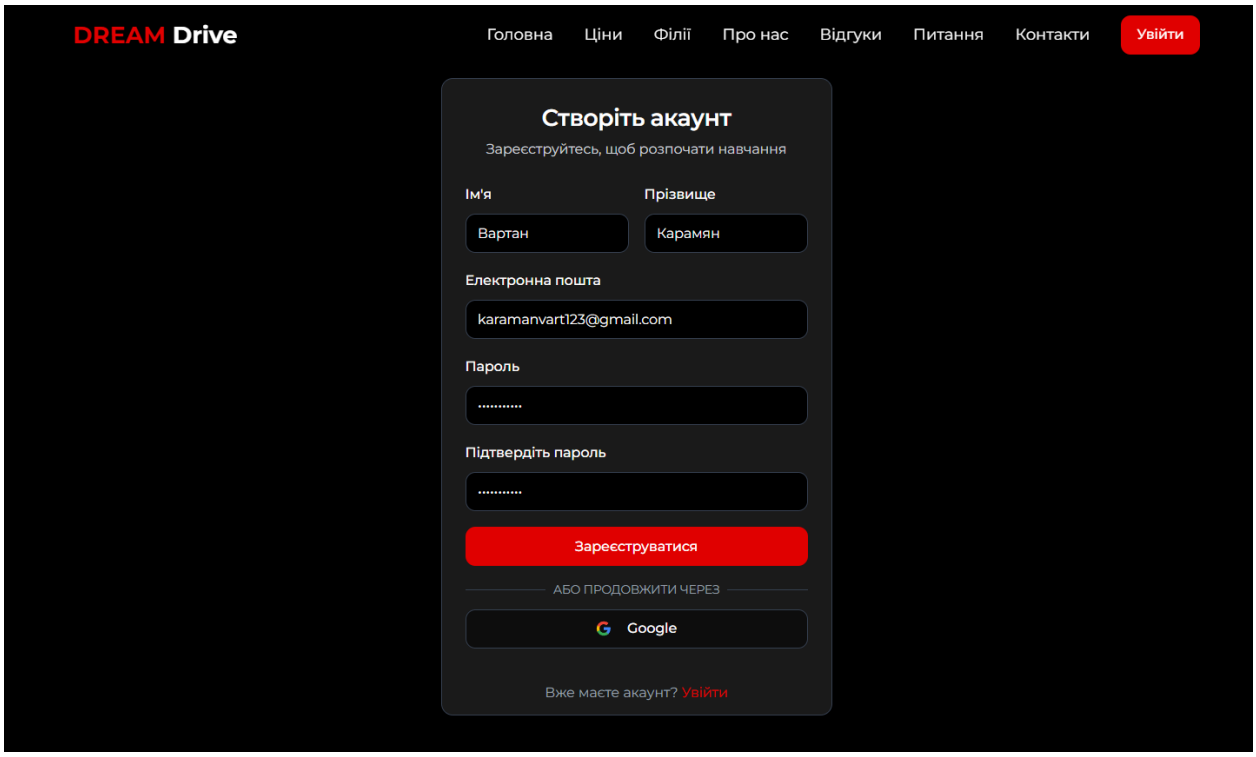


Рисунок 3.3 – Сторінка реєстрації

Також замість реєстрації можна обрати вхід за допомогою облікового запису Googleє (рисунок 3.4):

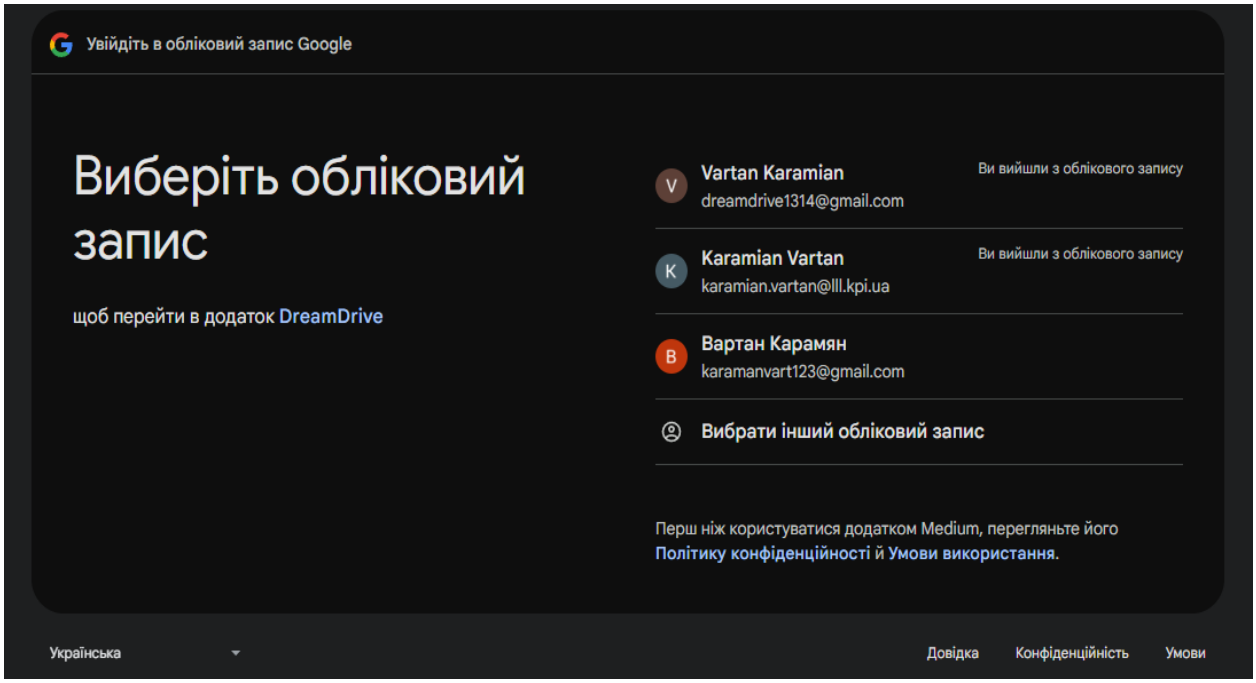


Рисунок 3.4 – Вхід за допомогою Google

При успішній реєстрації потрапляємо на панель керування студента, але поки що з обмеженим доступом (рисунок 3.5).

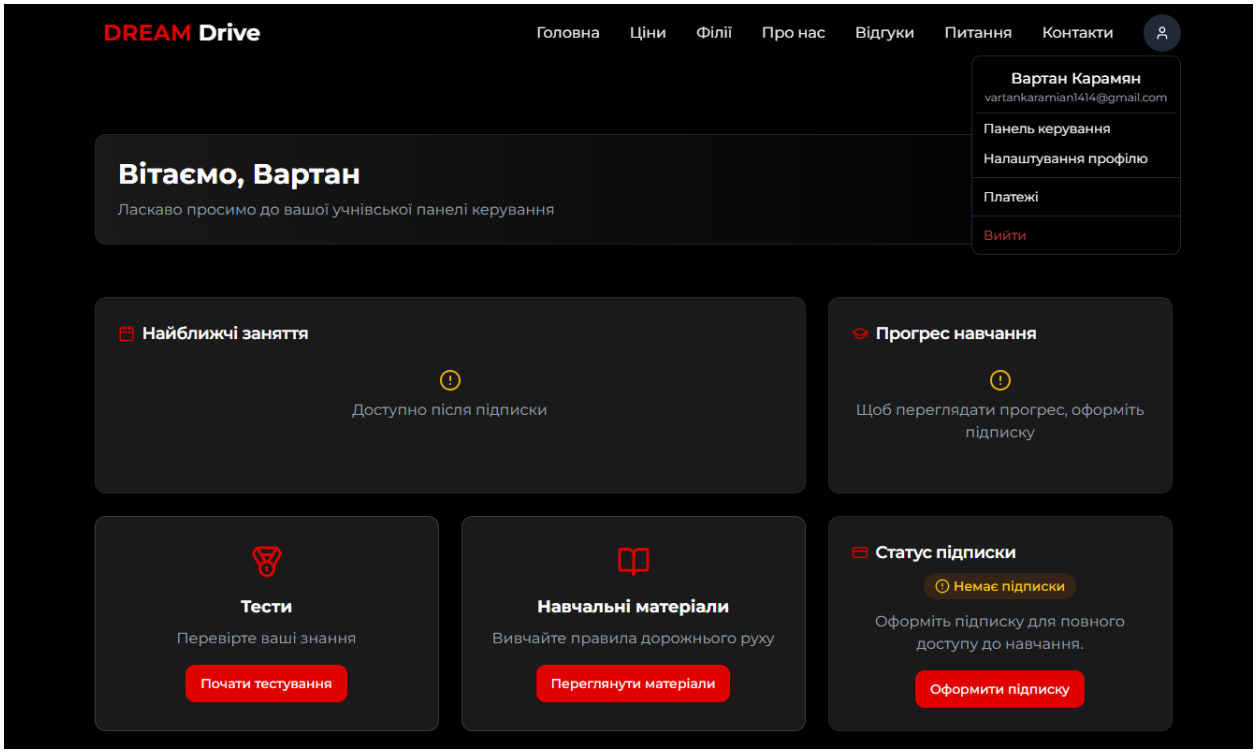


Рисунок 3.5 – Панель керування з обмеженим доступом

Щоб розпочати навчання переходимо в «Платежі» або натискаємр кнопку «Оформити підписку» (рисунок 3.6).

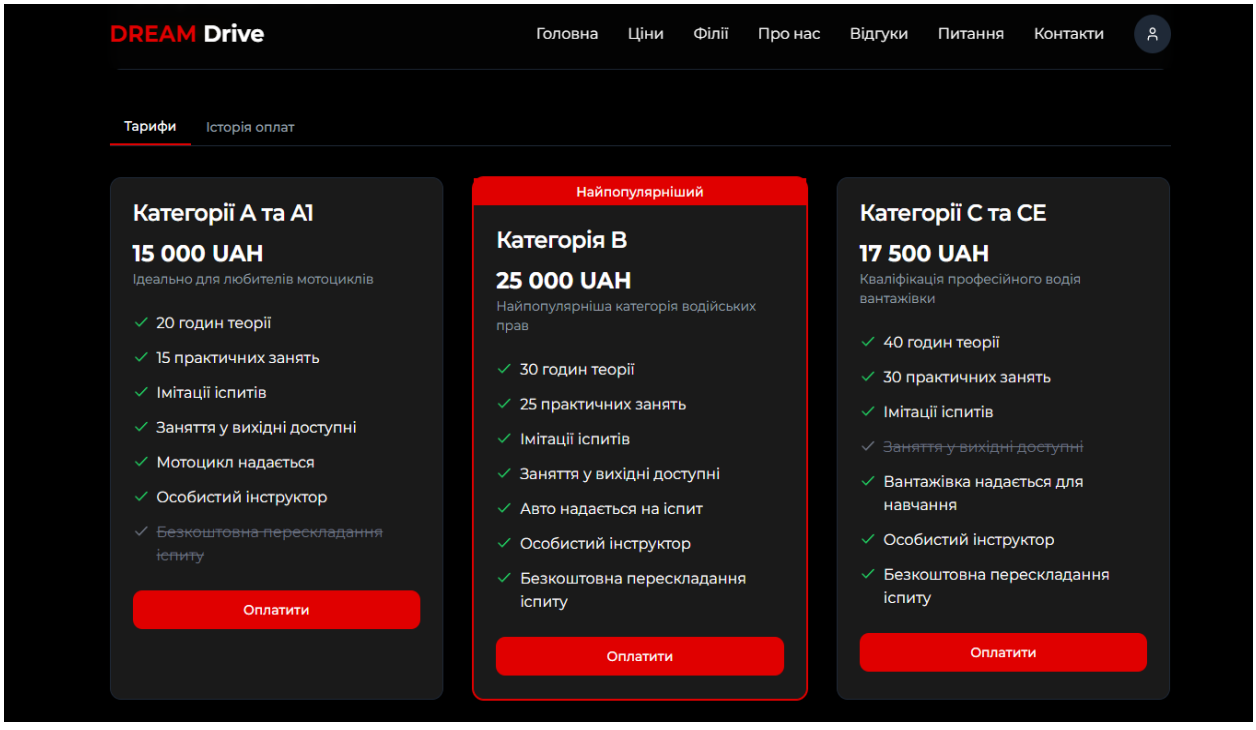


Рисунок 3.6 – Оформлення оплати

Натискаємо «Оплатити» та потрапляємо до сервісу оплат LiqPay, де можна обрати зручний спосіб та здійснити оплату (рисунок 3.7)

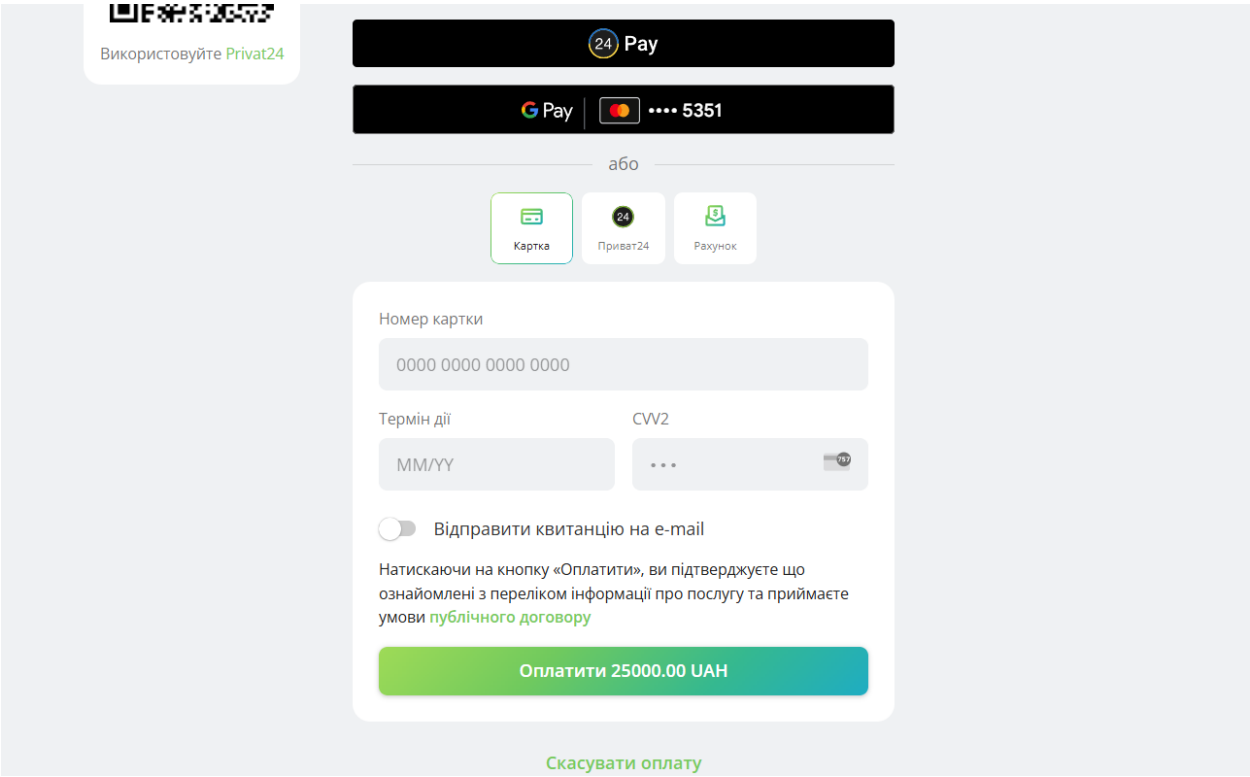


Рисунок 3.7 – Оплата через LiqPay

Проводимо оплату та натискаємо кнопку повернення до застосунку, де отримуємо повідомлення про статус оплати (рисунок 3.8).

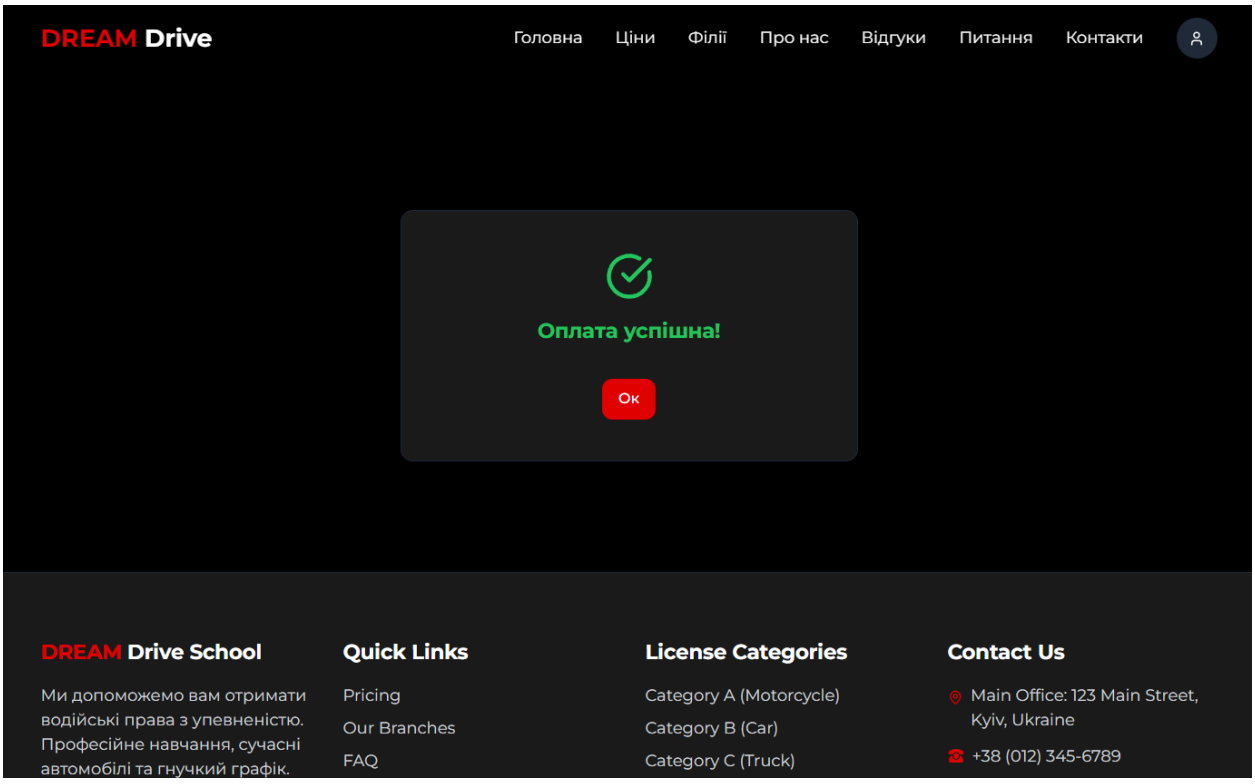


Рисунок 3.8 – Статус оплати

Повертаємося до платежів, де можемо переглянути історію оплат (рисунок 3.9).

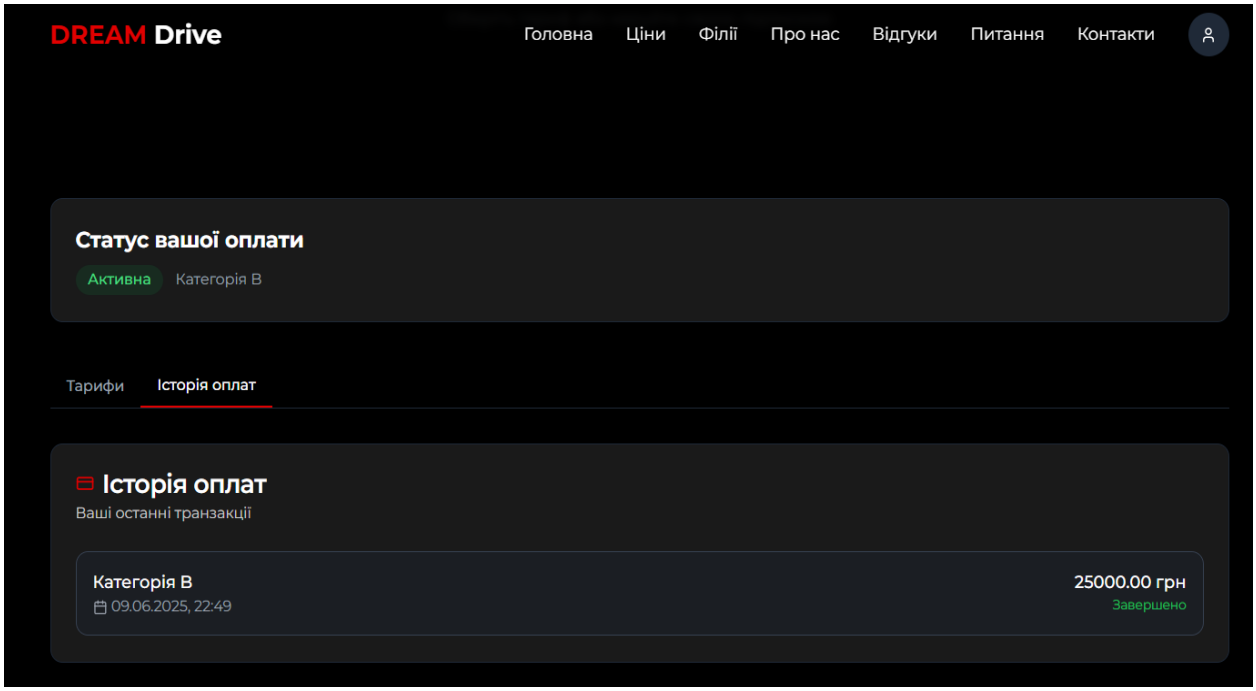


Рисунок 3.9 – Історія оплат

Отримуємо повний доступ до панелі керування (рисунок 3.10) та модулів: «Навчальні матеріли, «Мої заняття» та «Інтерактивні тести».

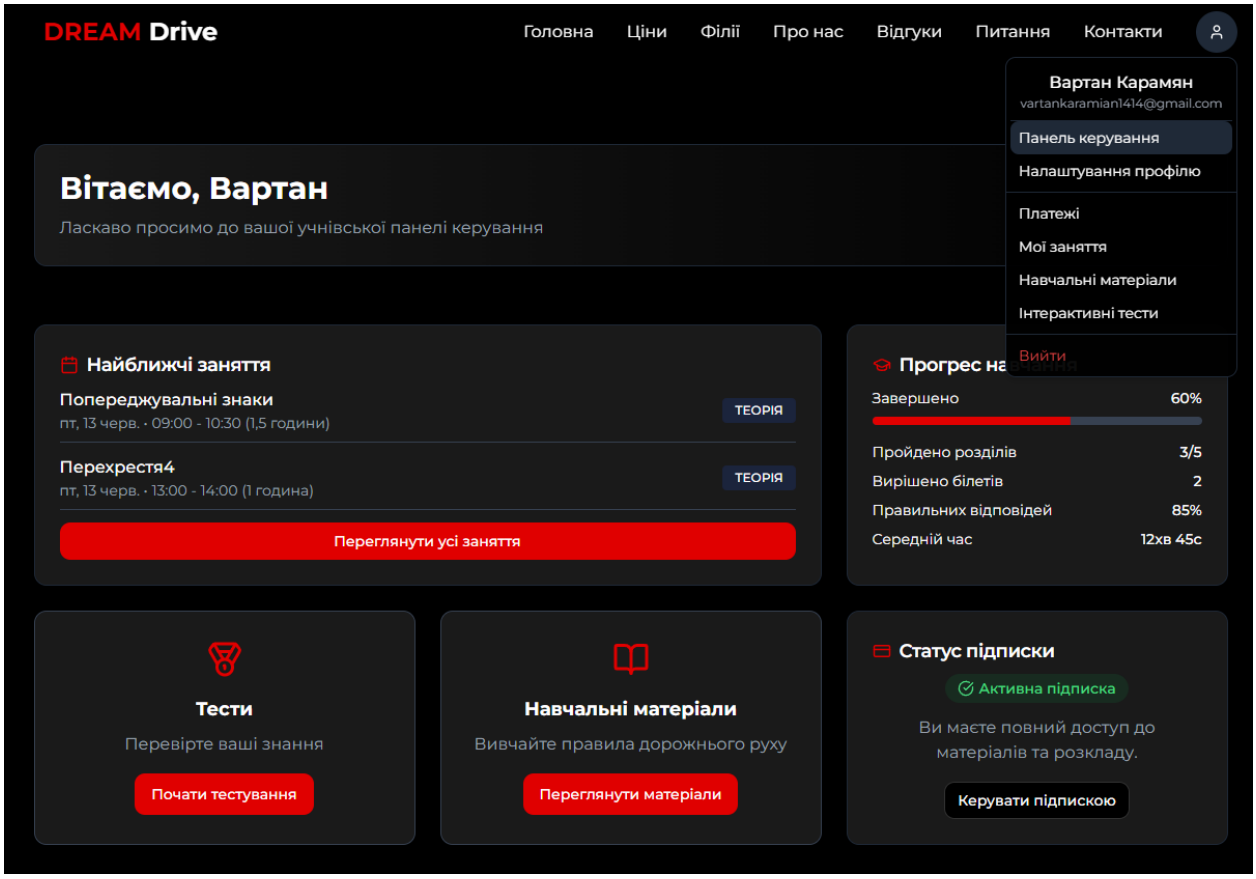


Рисунок 3.10 – Панель керування студента

На сторінці «Навчальні матеріали» обираємо потрібний розділ і тему ПДР та можемо з ними ознайомитися (рисунок 3.11).

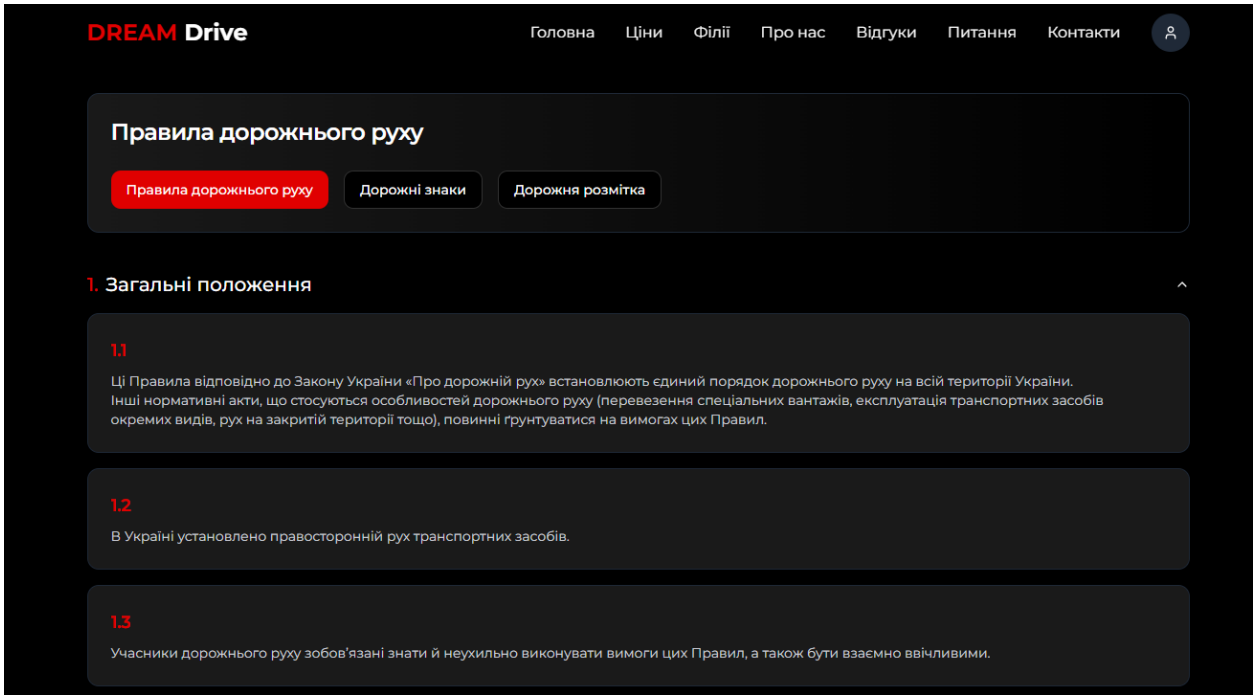


Рисунок 3.11 – Навчальні матеріали

На сторінці «Мої заняття» можемо забронювати практичне заняття та переглянути свої заплановані заняття (рисунок 3.12).

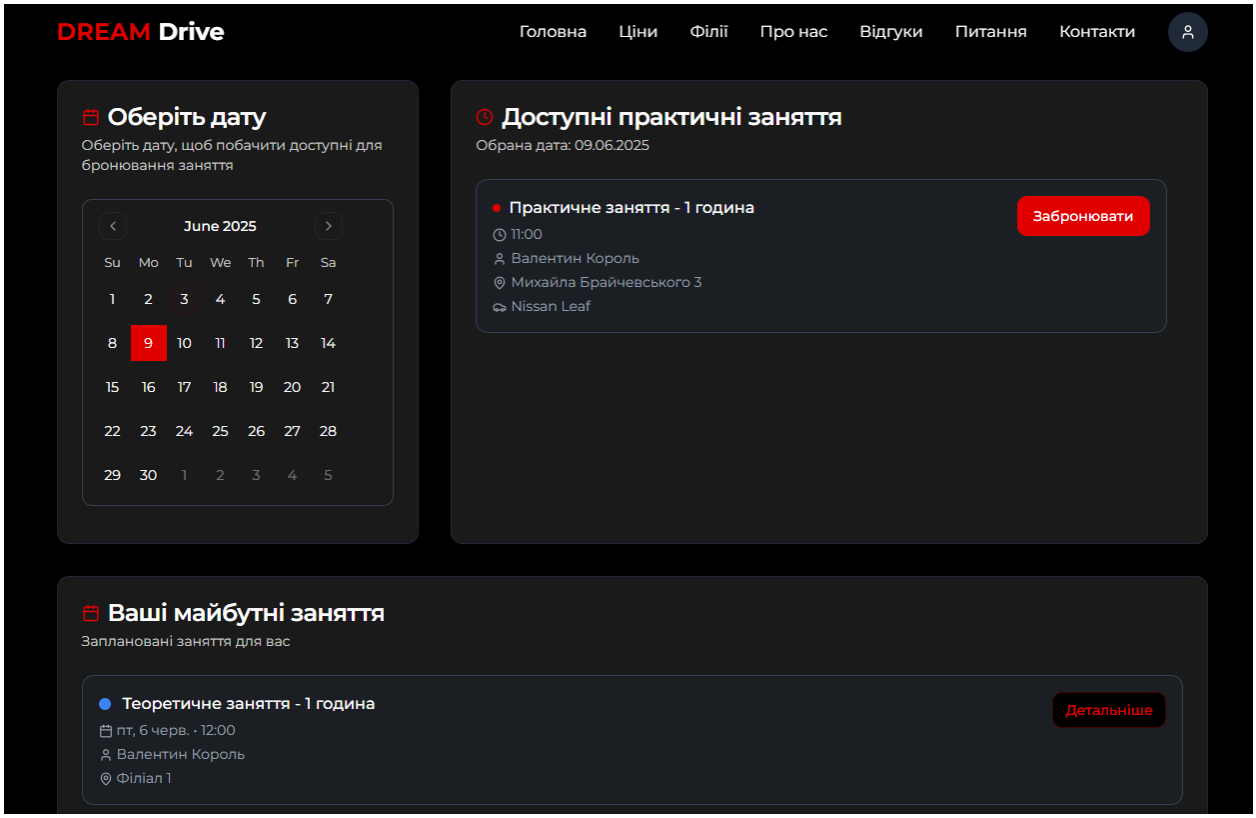


Рисунок 3.12 – Заняття студента

Щоб забронювати практичне заняття, потрібно обрати спочатку у календарі день, наприклад 12 червня та у правій частині з'являться вільні слоти для запису (рисунок 3.13).

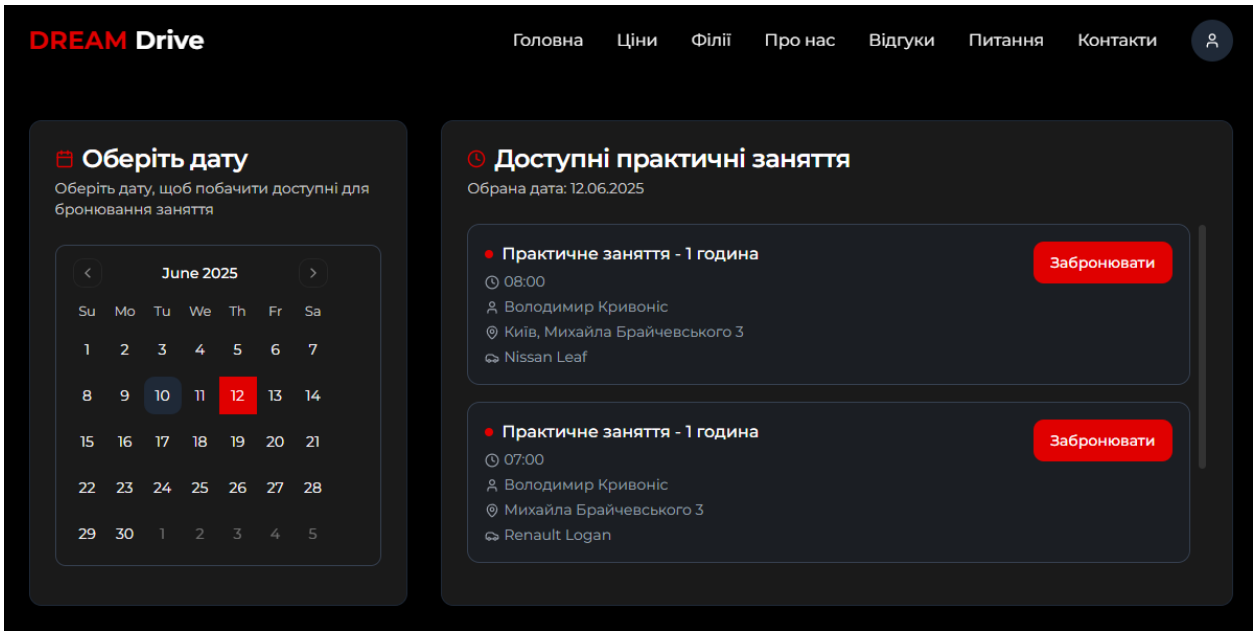


Рисунок 3.13 – Вибір дати бронювання

Тепер натискаємо кнопку бронювання та бачимо повідомлення, що у нас є заброньоване заняття (рисунок 3.14). Також зліва у нижньому кутку отримує сповіщення про успішне бронювання.

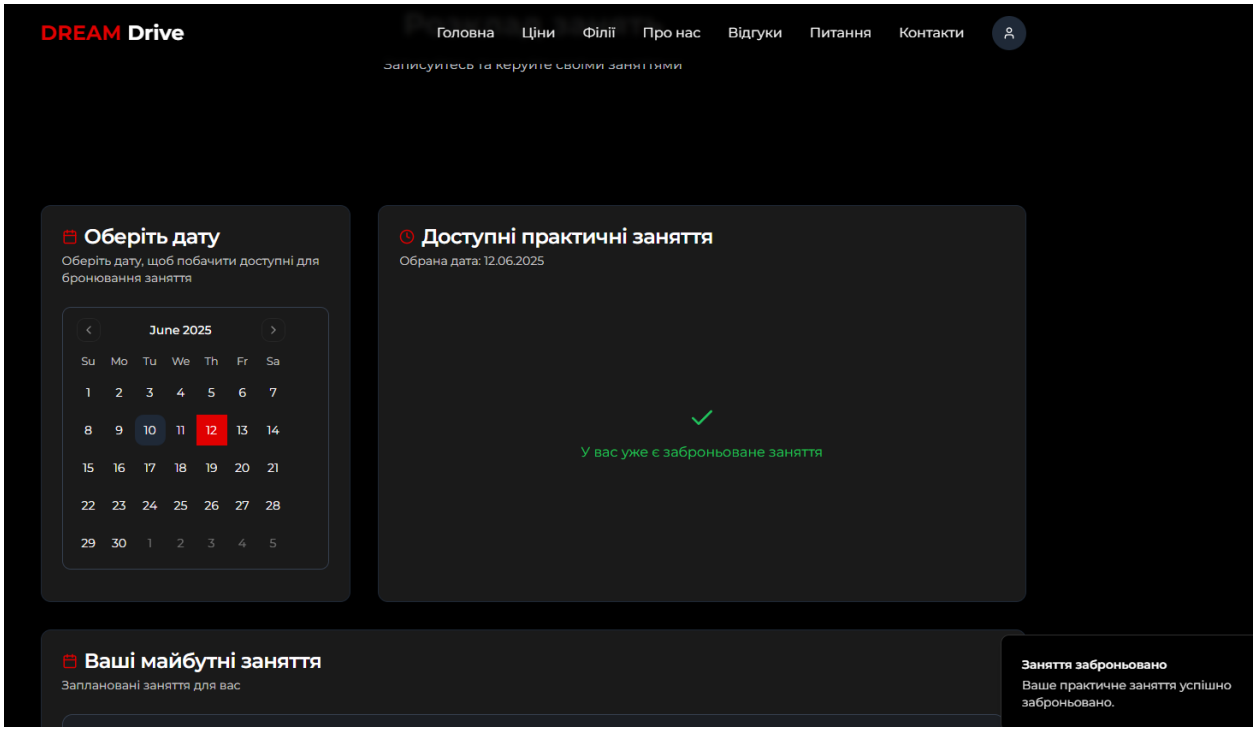


Рисунок 3.14 – Бронювання практичного заняття

Бачимо, що у секції запланованих занять з'явилося нове заброньоване заняття (рисунок 3.15).

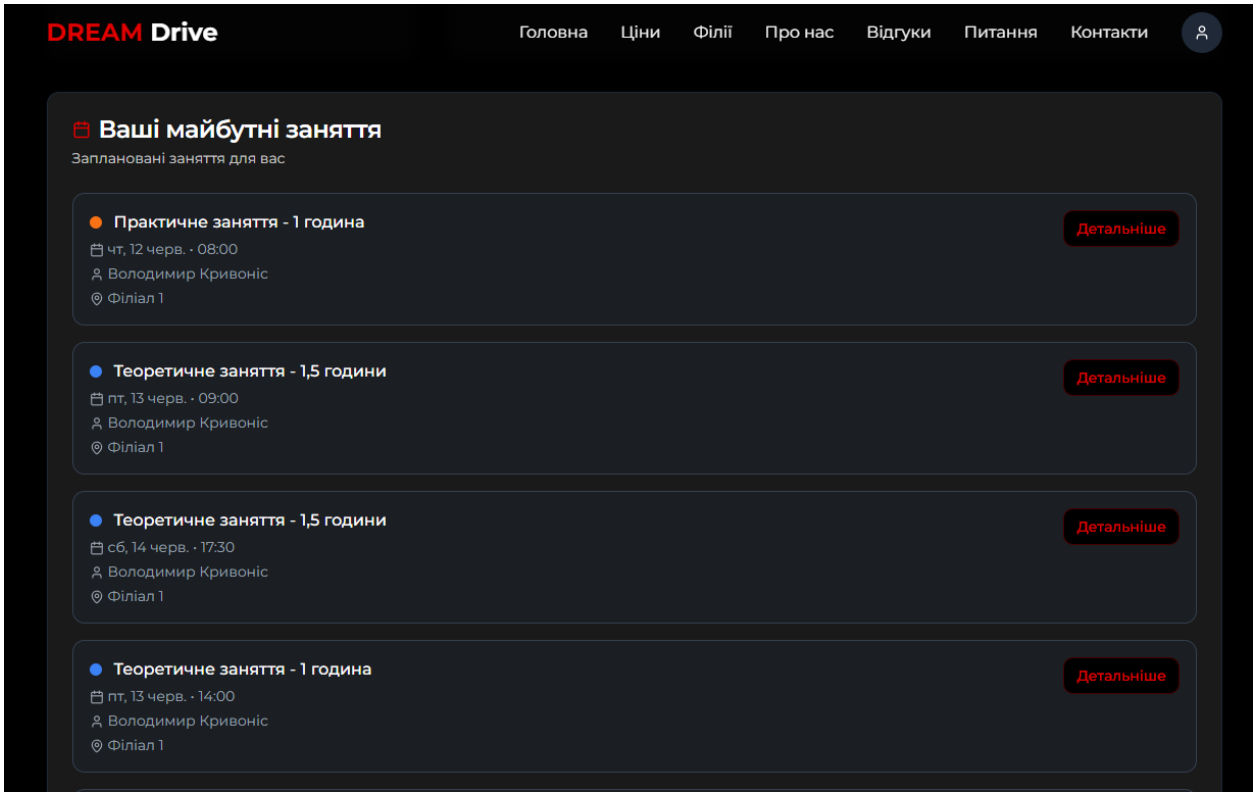


Рисунок 3.15 – Заплановані заняття

За допомогою кнопки «Детальніше» можемо переглянути додаткову інформацію про заняття (рисунок 3.16).

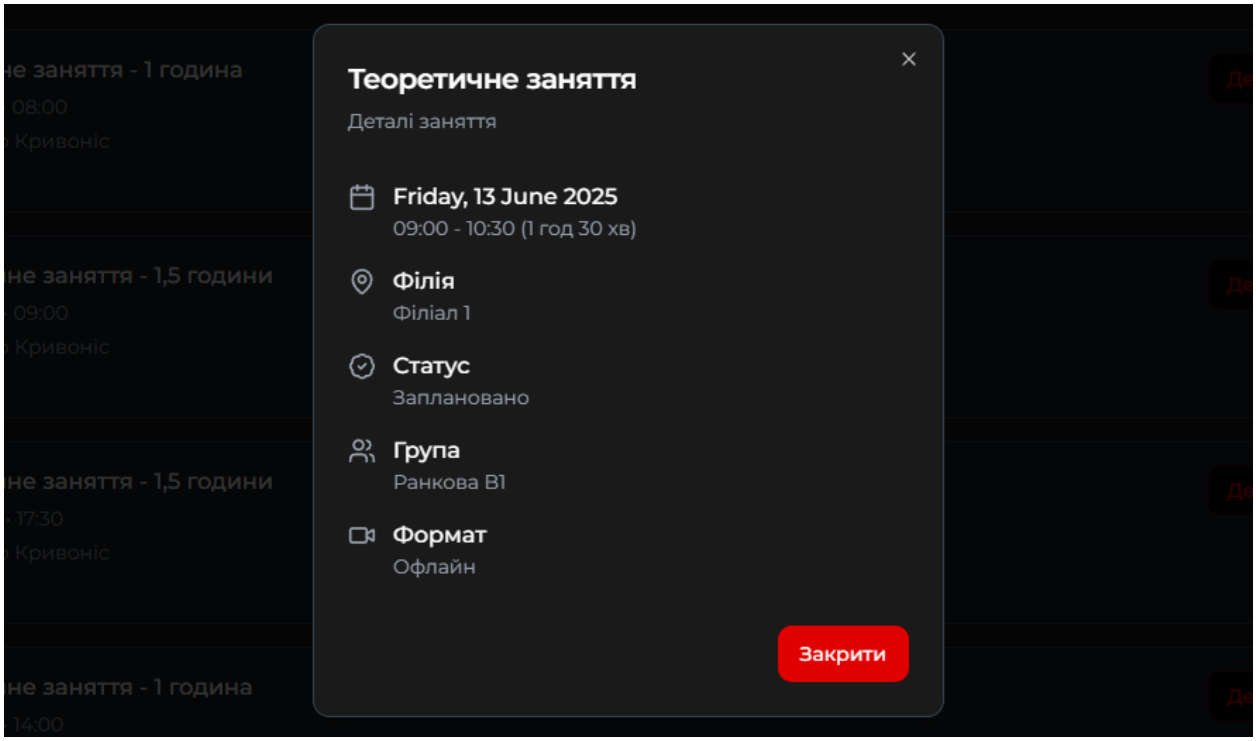


Рисунок 3.16 – Деталі заняття

На вкладці з тестами у верхній секцій можна обрати тип тесту та почати тестування (рисунок 3.17).

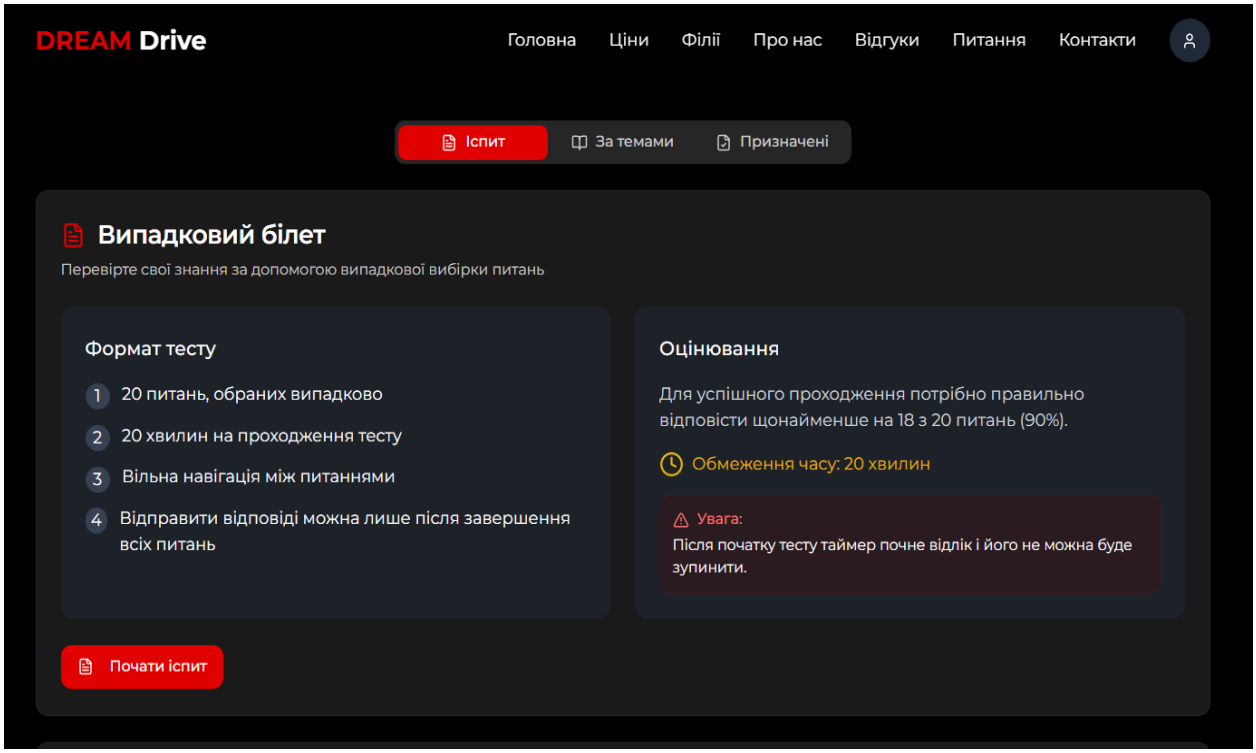


Рисунок 3.17 – Інтерактивні тести

Якщо натиснути «Почати іспит», то побачимо вікно з інформацією про іспит, де потрібно перед початком натиснути «Почати тест» (рисунок 3.18).

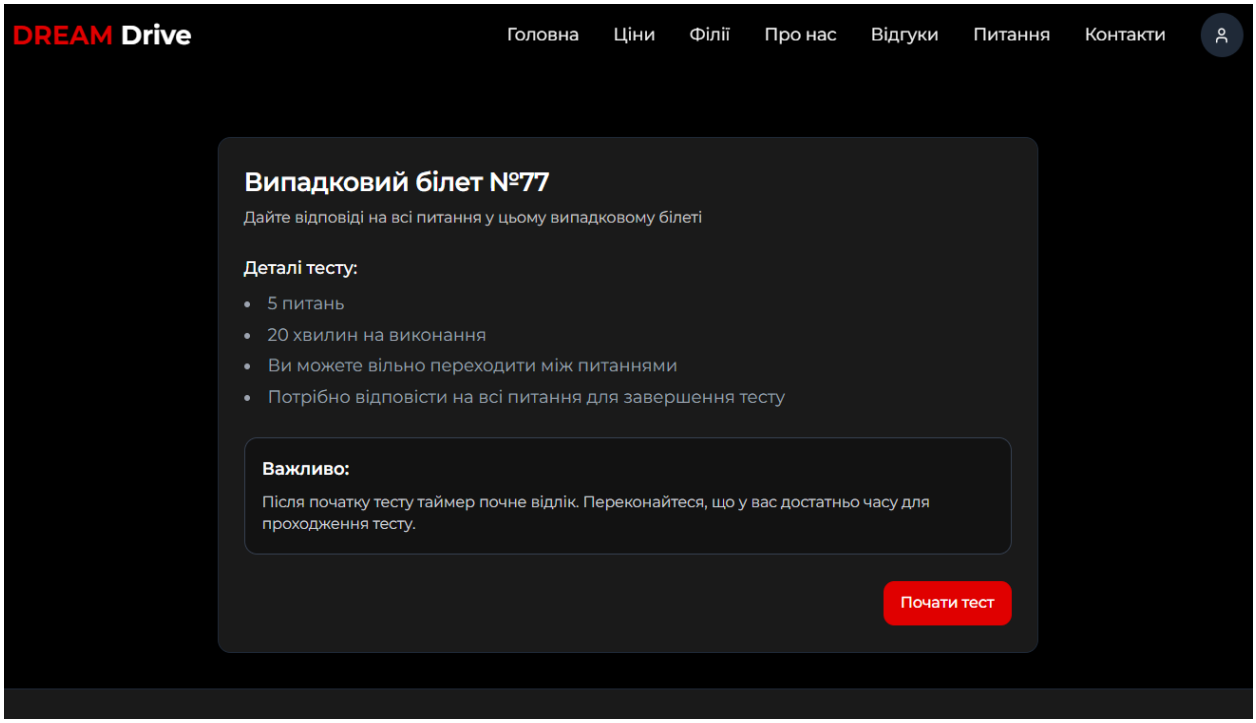


Рисунок 3.18 – Початок тесту

Далі можемо пройти тест послідовно відповідаючи на питання. Для перемикання між питанянми використовуємо кнопки навігації у нижній частині екрану (рисунок 3.19). Відповівши на всі питання завершуємо тест (рисунок 3.20).

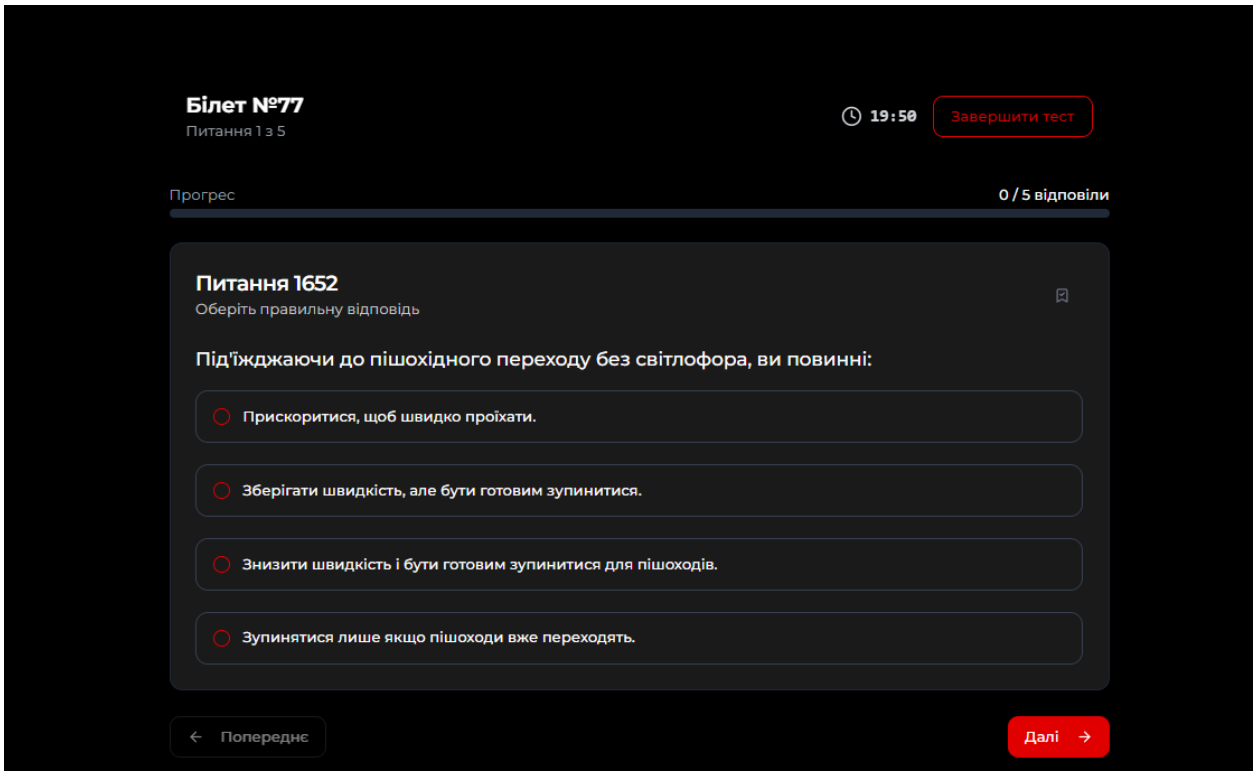


Рисунок 3.19 – Проходження тесту

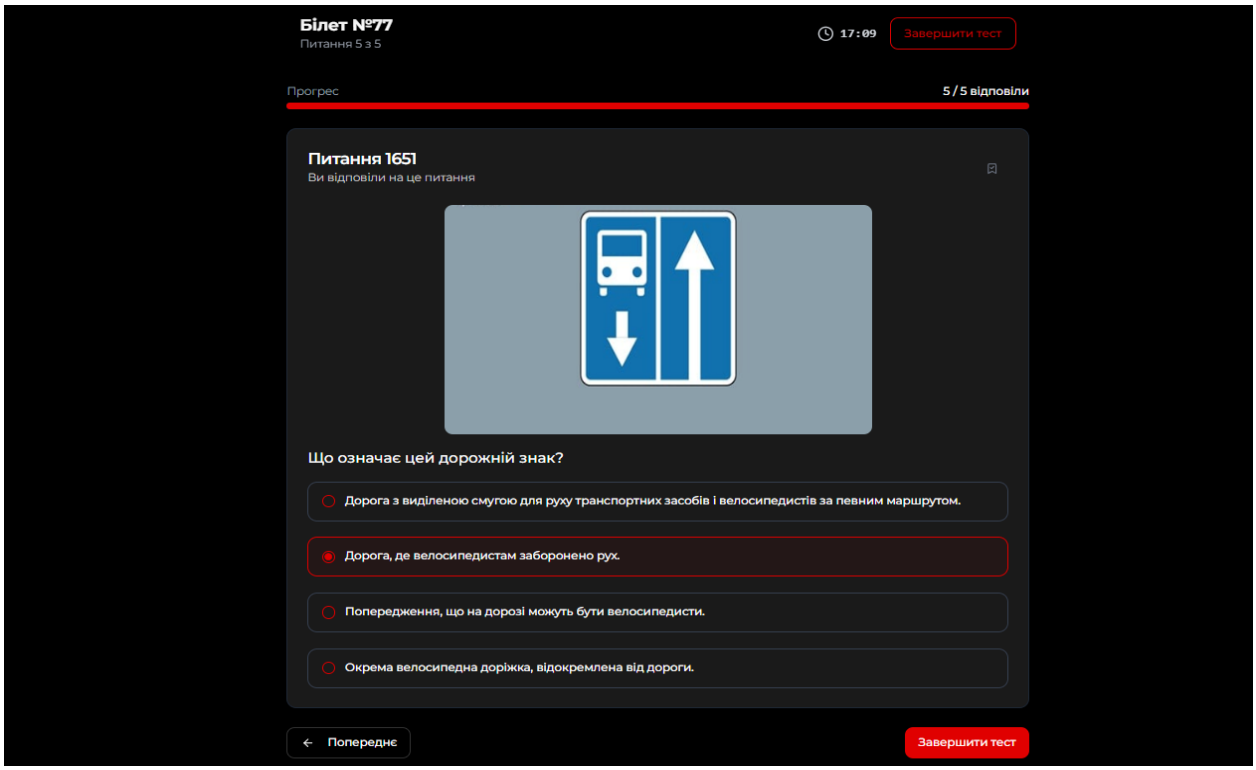


Рисунок 3.20 – Завершення тесту

Після завершення отримуємо результат тесту (рисунок 3.21) та детальний розбір усіх питань (рисунок 3.22). За допомогою кнопки До тестів можемо повернутися на сторінку «Інтерактивні тести».

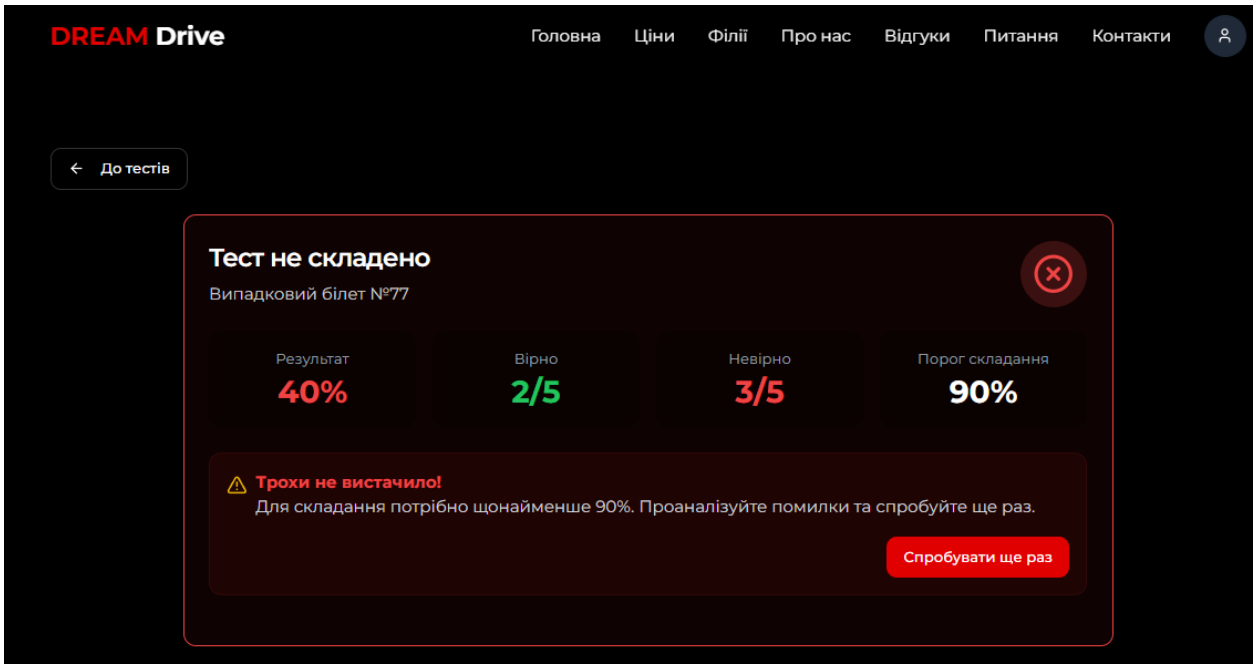


Рисунок 3.21 – Результат тесту

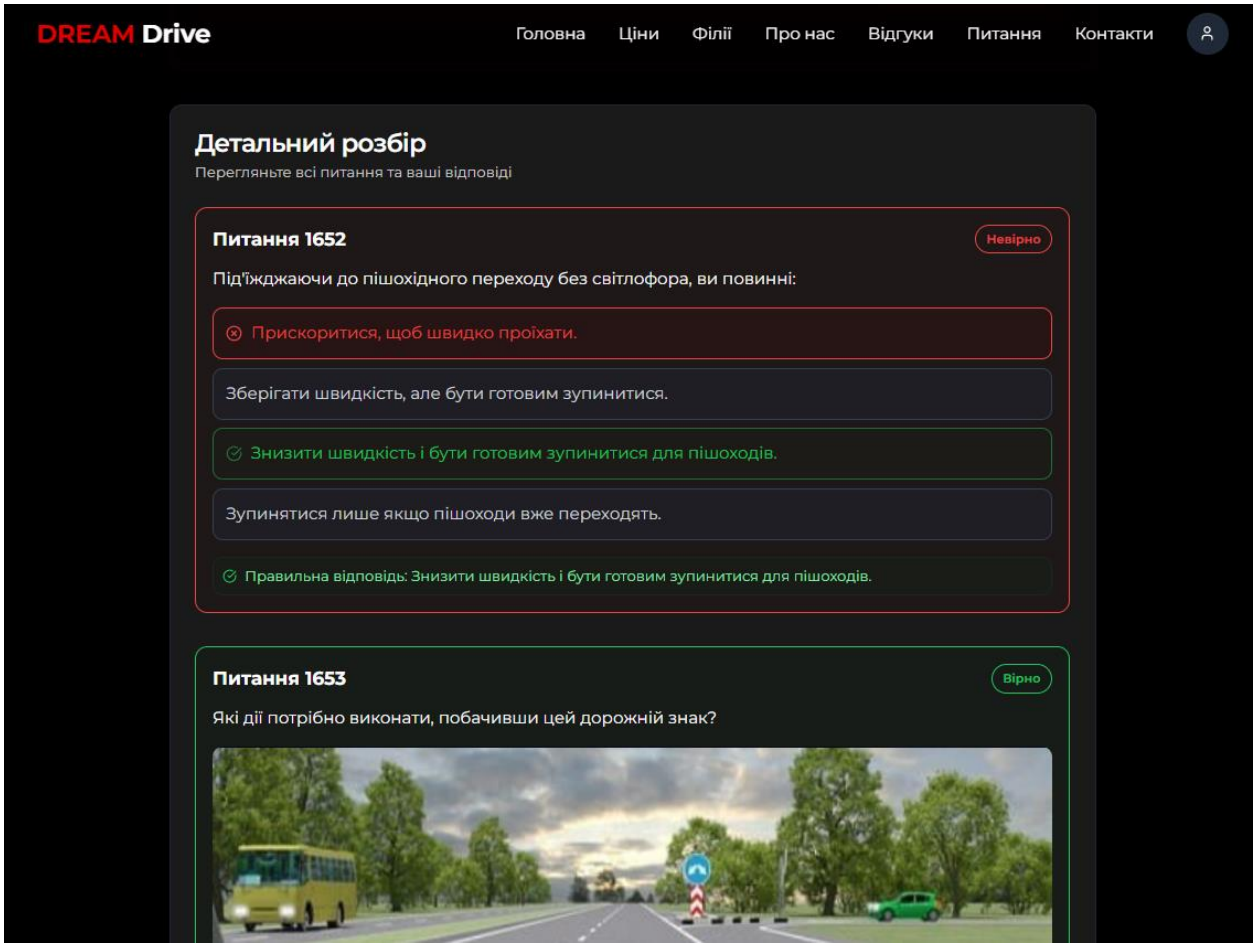


Рисунок 3.22 – Детальний розбір тесту

У нижній частині сторінки «Інтерактивні тести» можна переглянути результати своїх тестів (рисунок 3.23). Щоб переглягути усю історію внизу для цього є відповідна кнопка (рисунок 3.24).

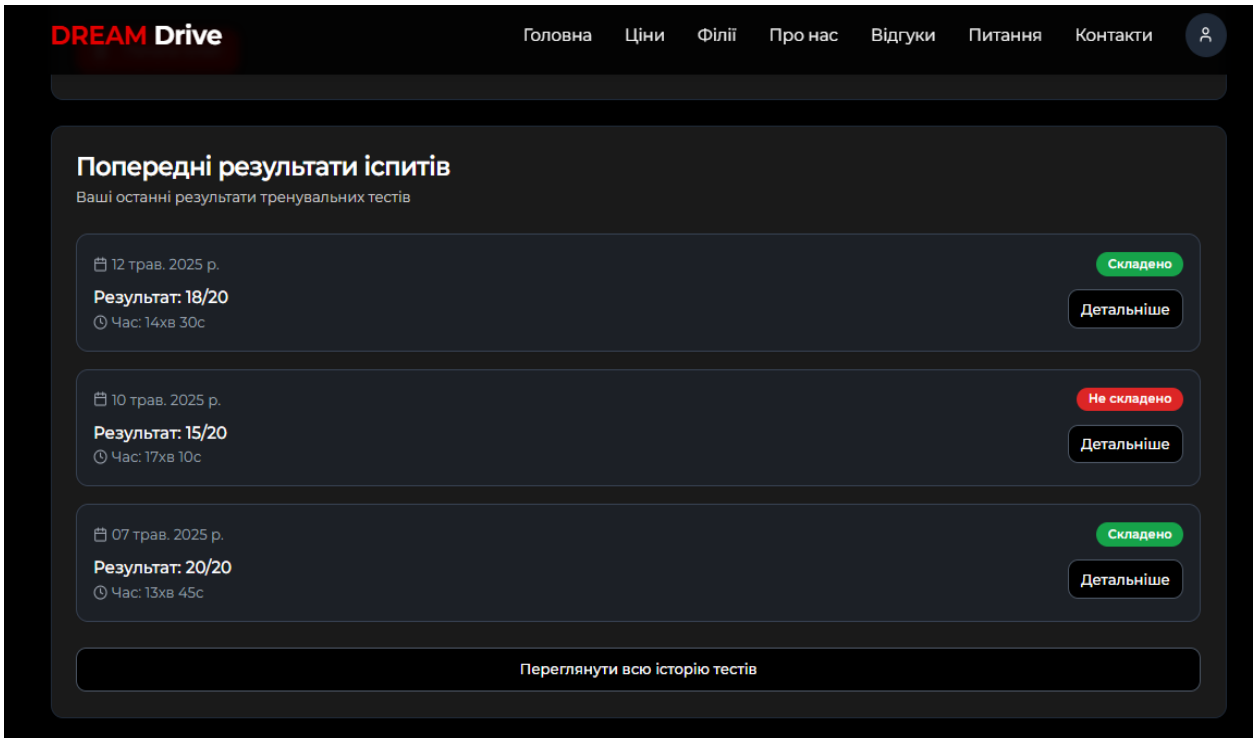


Рисунок 3.23 – Результати тестів

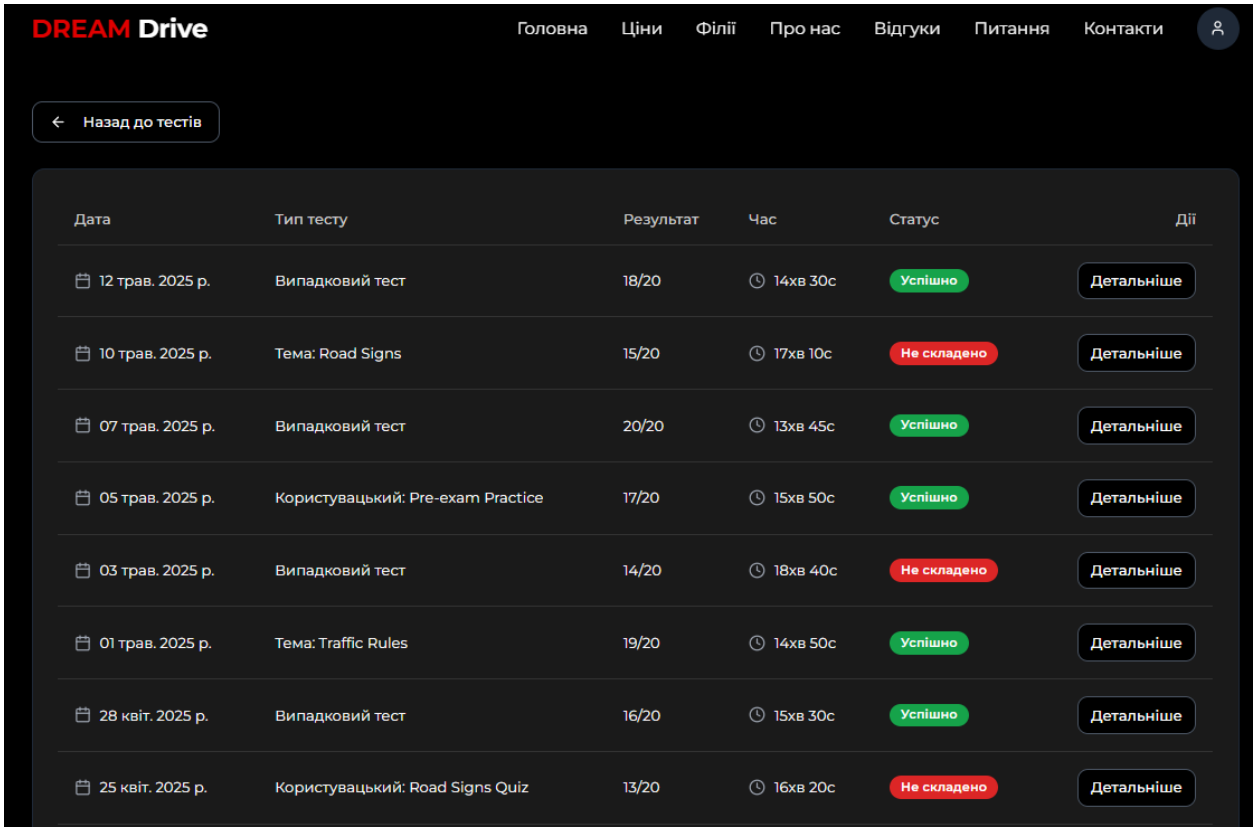


Рисунок 3.24 – Повна історія результатів тестів

Щоб переглянути деталі пройденого тесту, натискаємо «Детальніше» (рисунок 3.25).

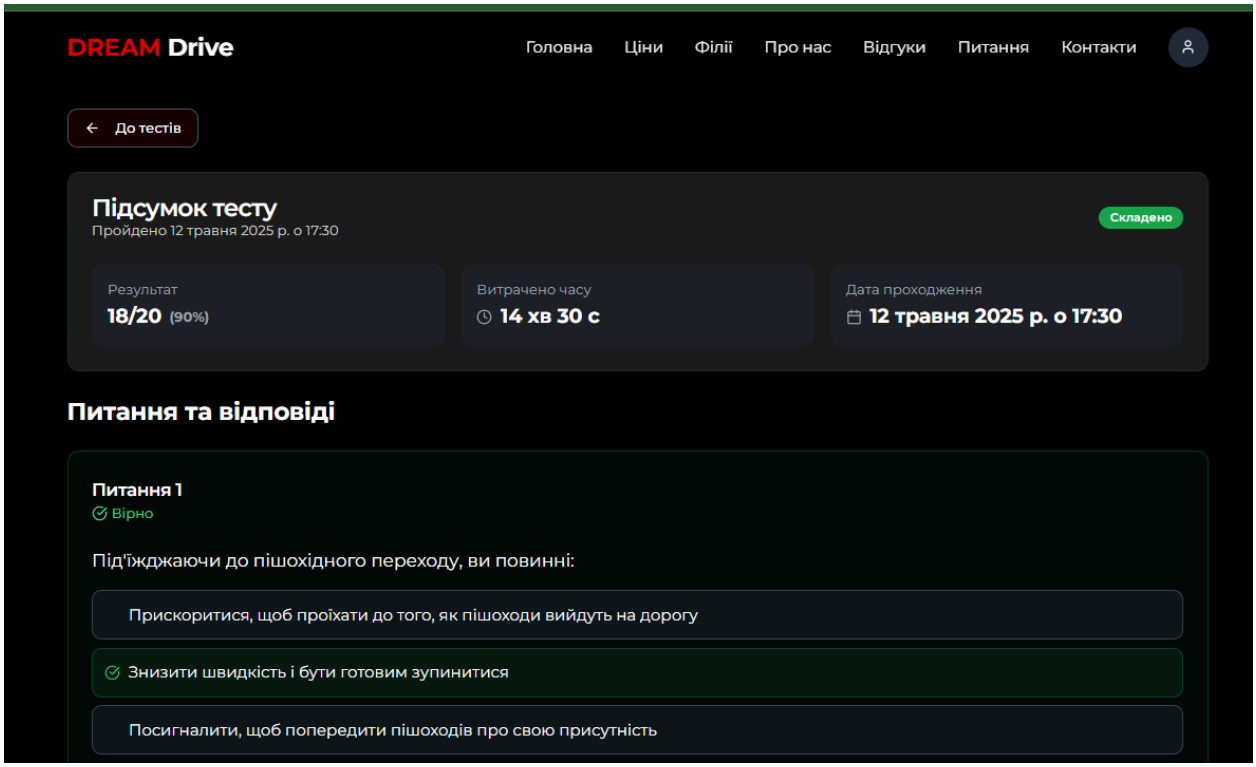


Рисунок 3.25 – Деталі результату тесту

Аналогічно можна проходити тести за темами та призначені тести, якщо обрати відповідний тип тесту. Оберемо тест «За темами» (рисунок 3.26).

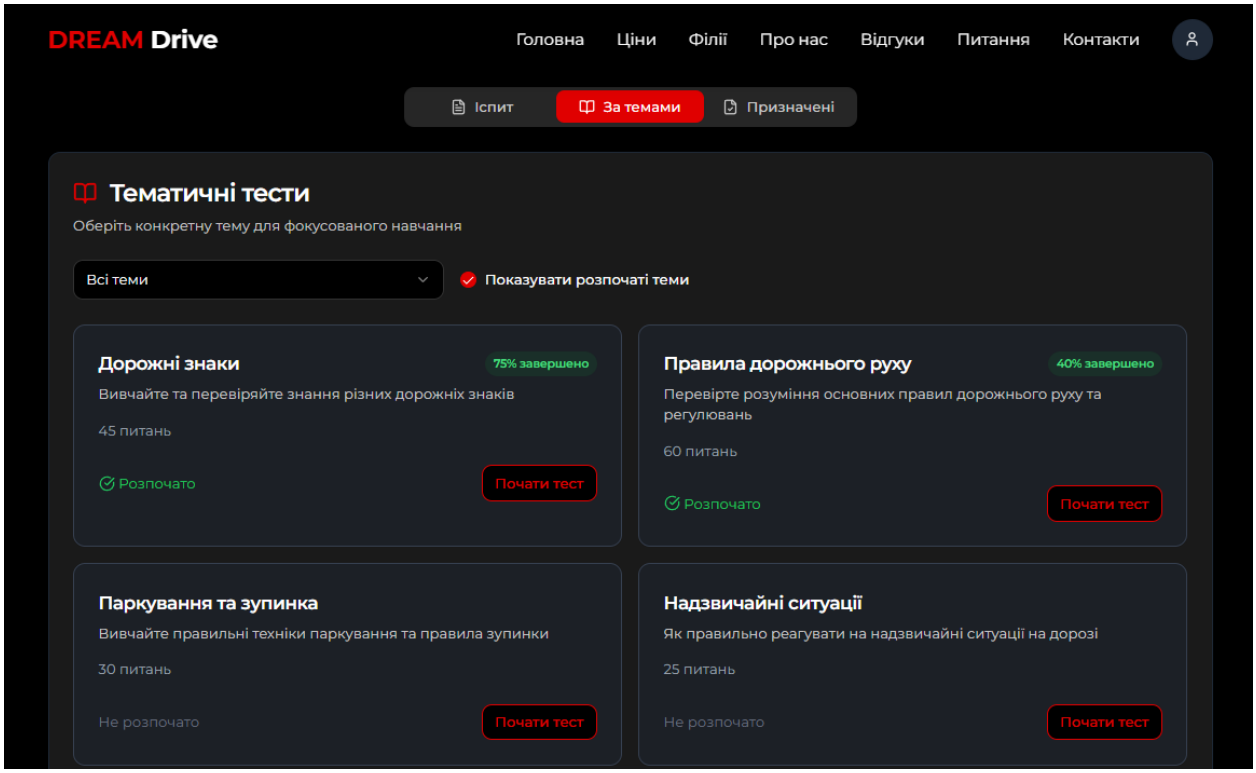


Рисунок 3.25 – Тест за темами

3.2 Викладач

Тепер розглянемо сценарій для користувача, який авторизувався під роллю викладача. При вході потрапляємо на панель керування викладача (рисунок 3.26), де можемо перейти до груп викладача (рисунок 3.27).

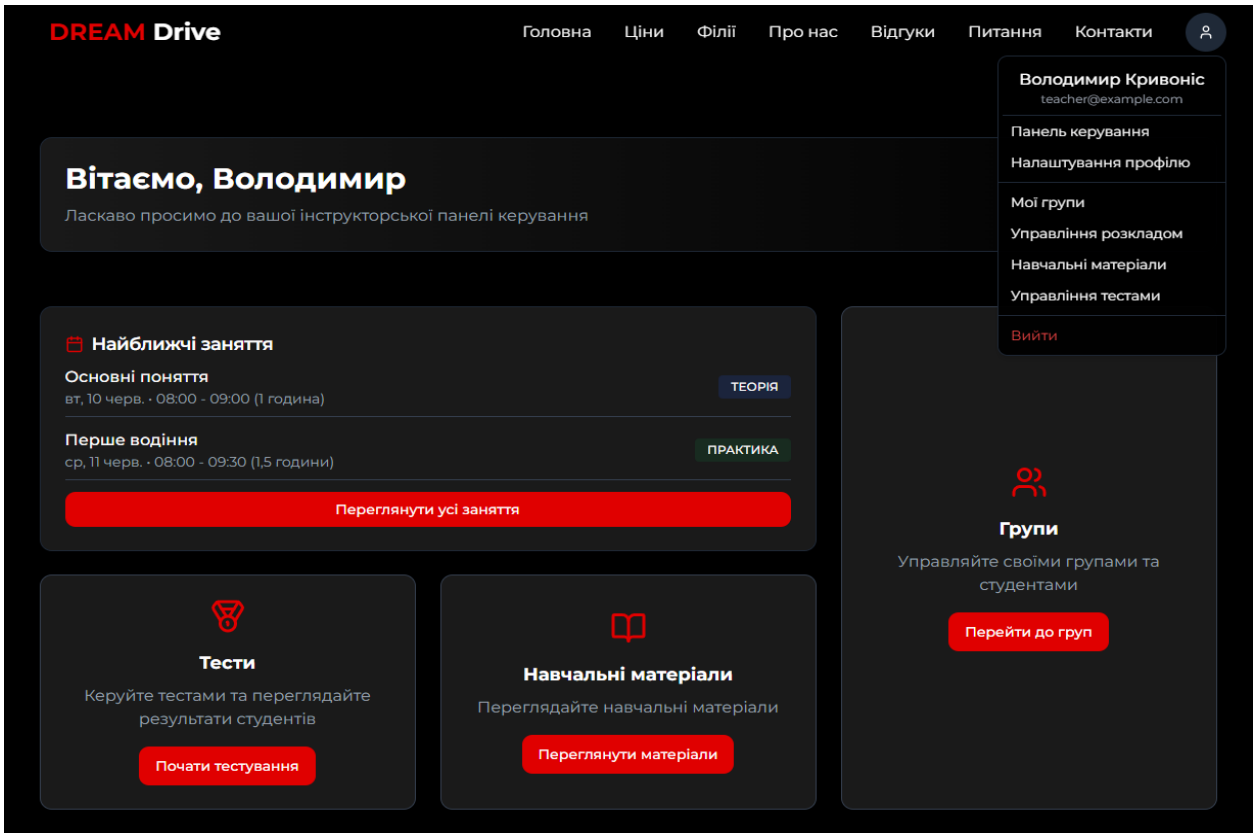


Рисунок 3.26 – Панель керування викладача

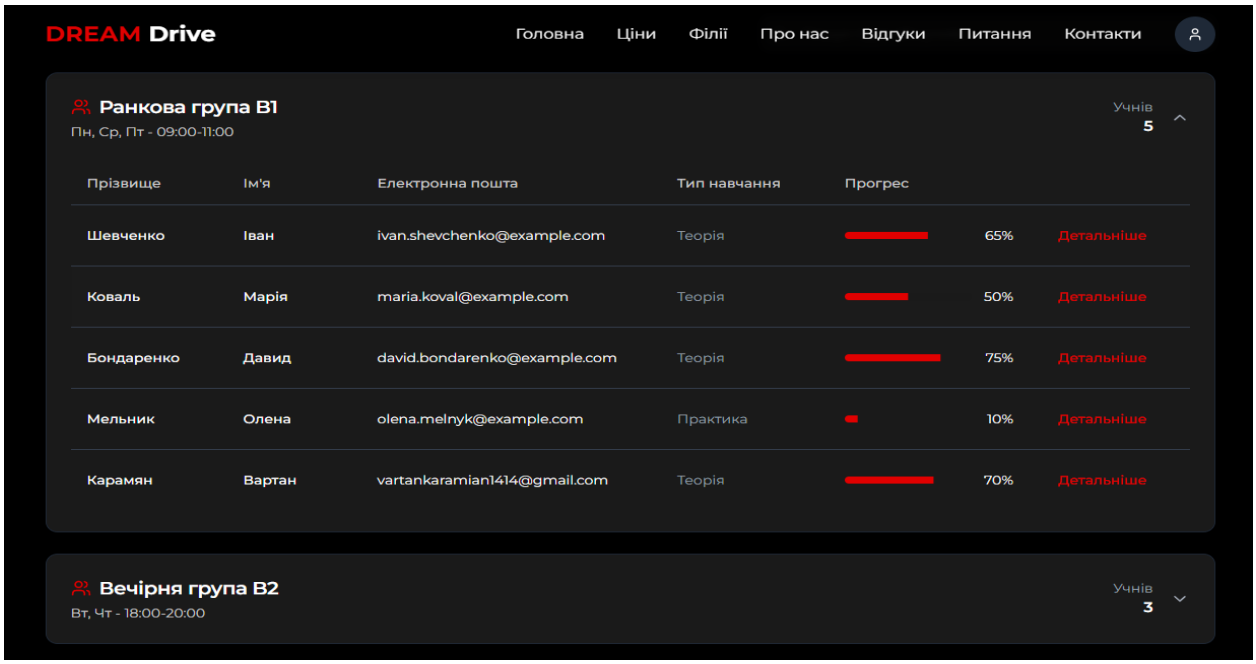


Рисунок 3.27 – Сторінка груп викладача

Перейдемо тепер до вкладки «Розклад занять» та оберемо вікно «Мої заняття». Тут ми можемо побачити список найближчих занять (рисунок 3.28)

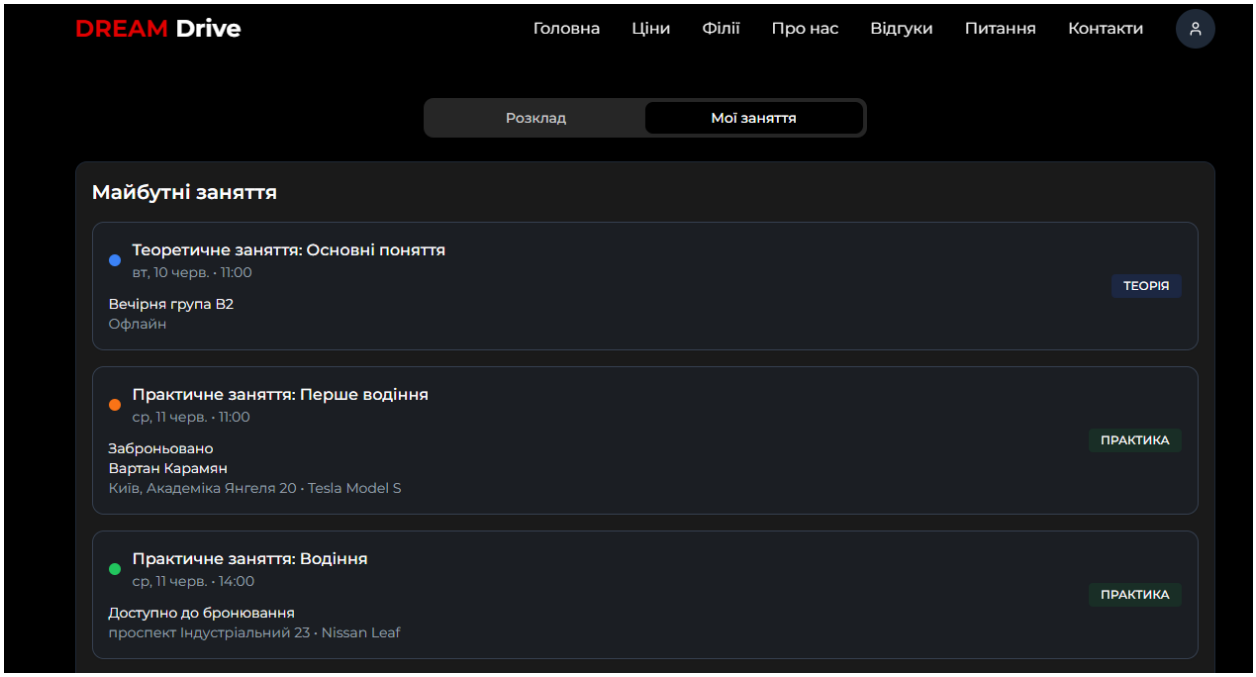


Рисунок 3.28 – Сторінка «Мої заняття» для викладача

У вікні «Розклад» доступний інтерактивний календар з можливістю перегляду заповнених занять (рисунок 3.29).

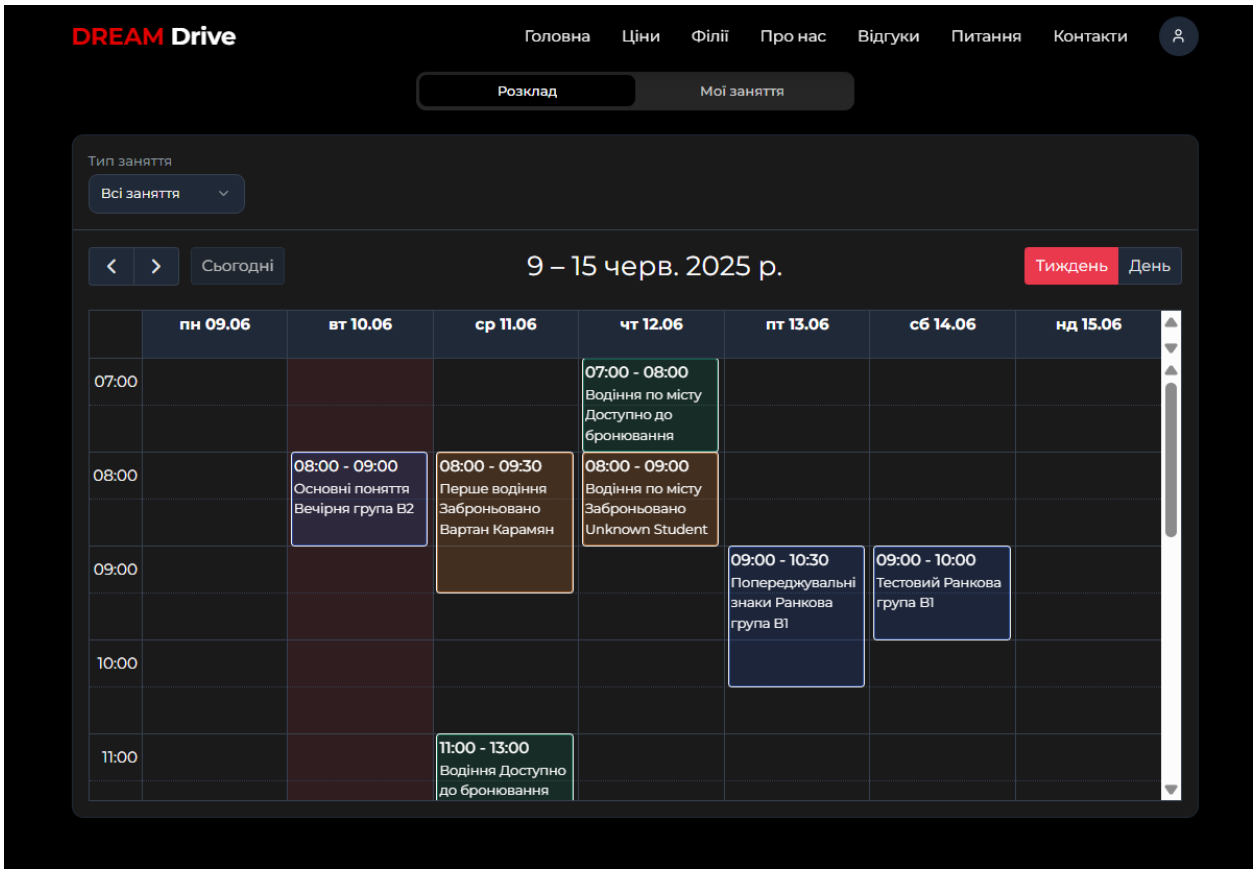


Рисунок 3.29 – Інтерактивний календар викладача

Якщо натиснути на одне з занять, то відкриється вікно з деталями заняття (рисунок 3.30).

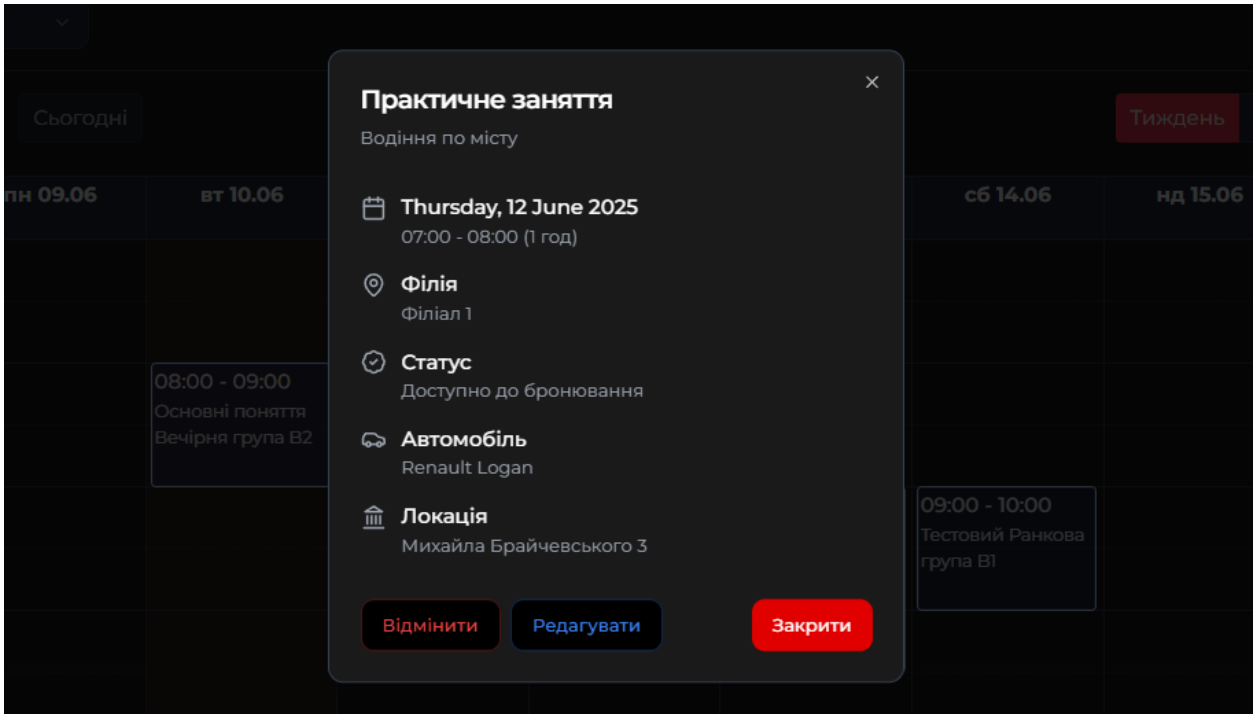


Рисунок 3.30 – Деталі заняття

У цьому вікні можемо відмінити або відредагувати заняття. Якщо натиснемо «Редагувати», то відкриється форма редагування, де можна внести зміни та зберегти або скасувати їх (рисунок 3.31).

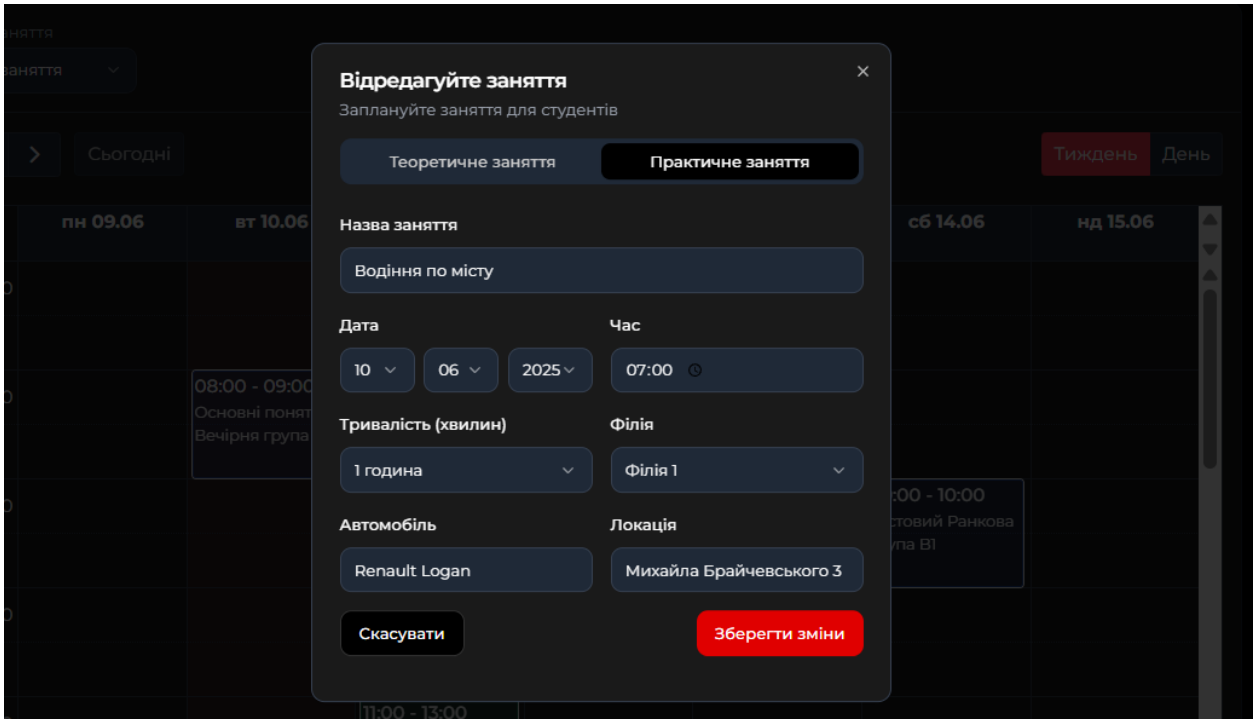


Рисунок 3.31 – Редагування заняття

Щоб додати нове заняття потрібно натиснути в календарі на вільний таймслот та з’явиться форма створення нового заняття (рисунок 3.32).

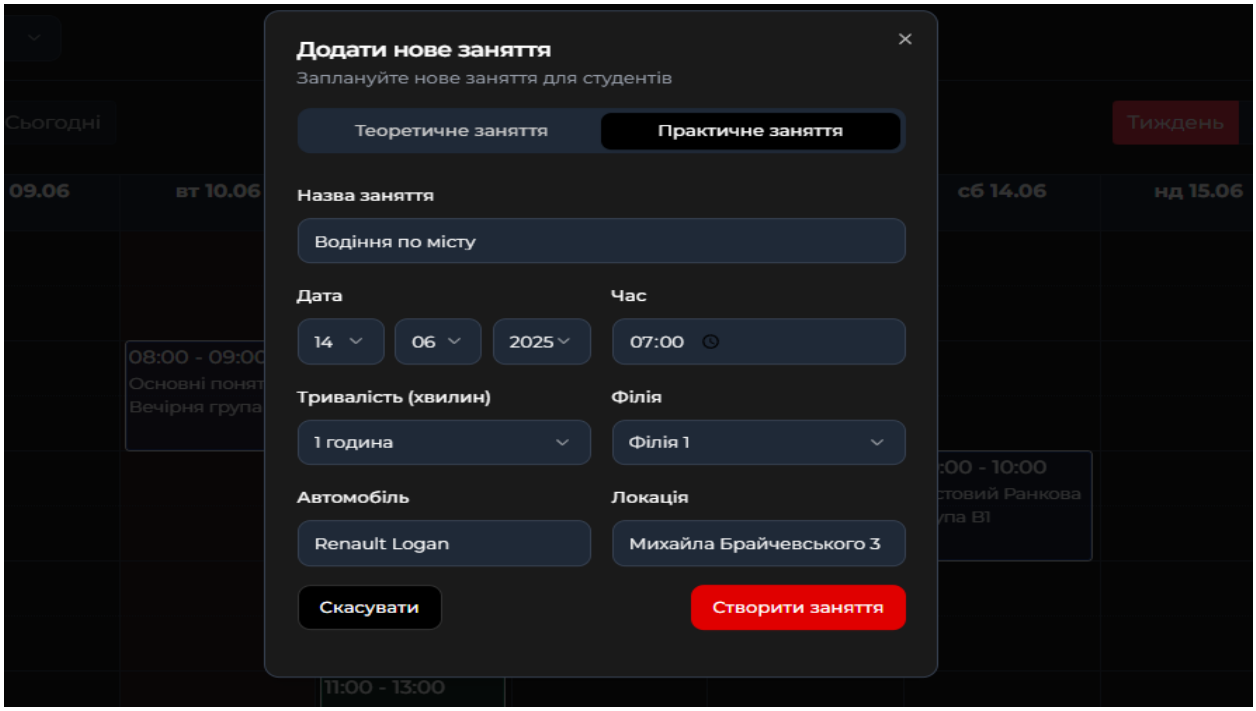


Рисунок 3.32 – Створення нового заняття

На сторінці з тестами можна переглянути опубліковані тести та статистику по ним (рисунок 3.33).

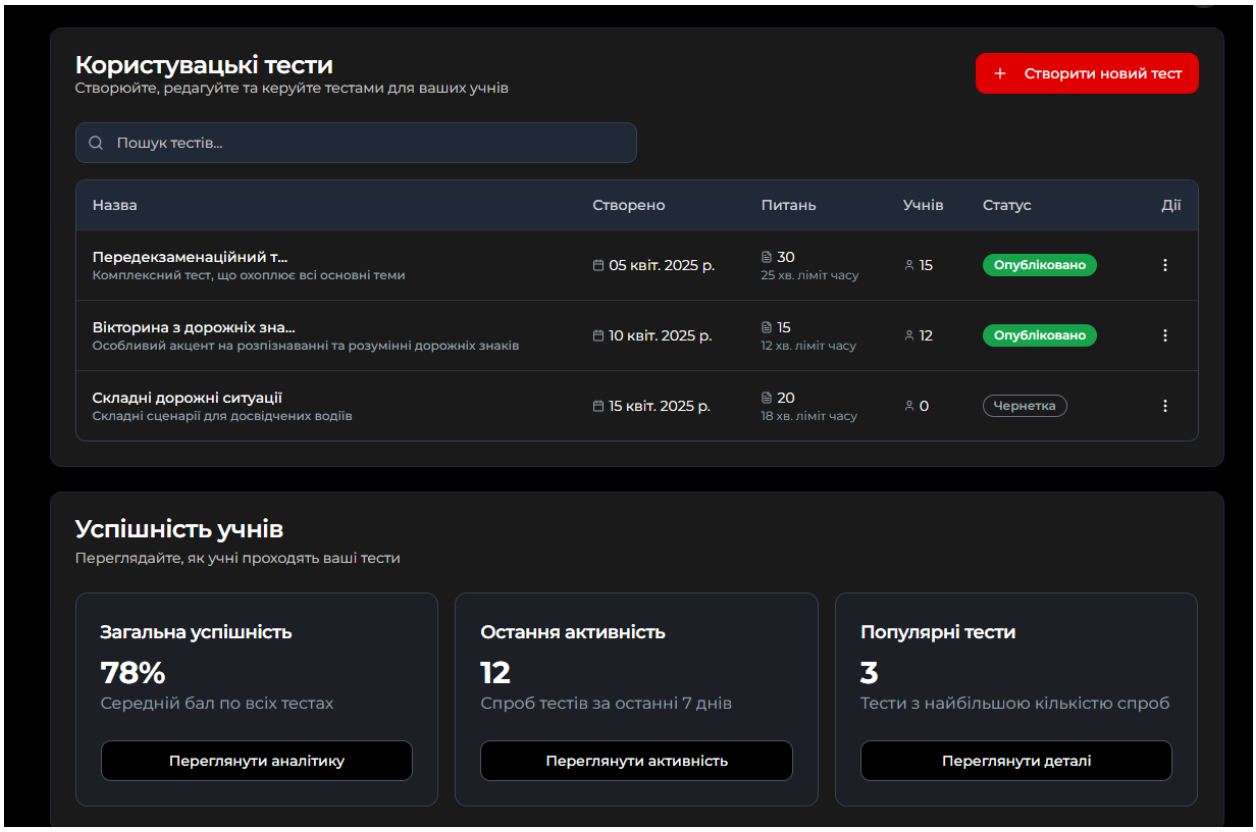
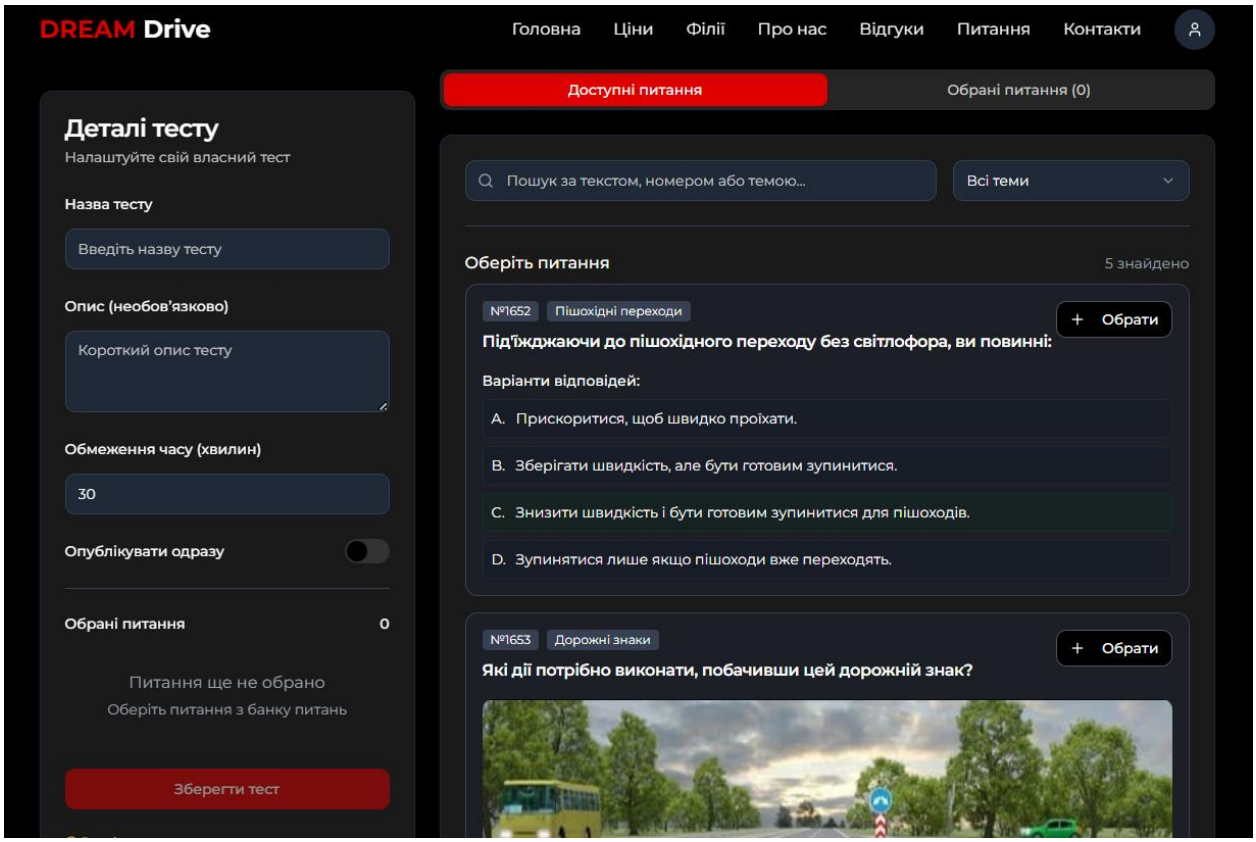


Рисунок 3.33 – Тести викладача

Якщо натиснути «Створити новий тест», то відкриється вікно створення тесту (рисунок 3.34). Тут потрібно заповнити інформацію про тест та обрати питання з доступних. Коли все готово натискаємо «Зберегти тест».



Також можемо перегляти навчальні матеріали (рисунок 3.35).

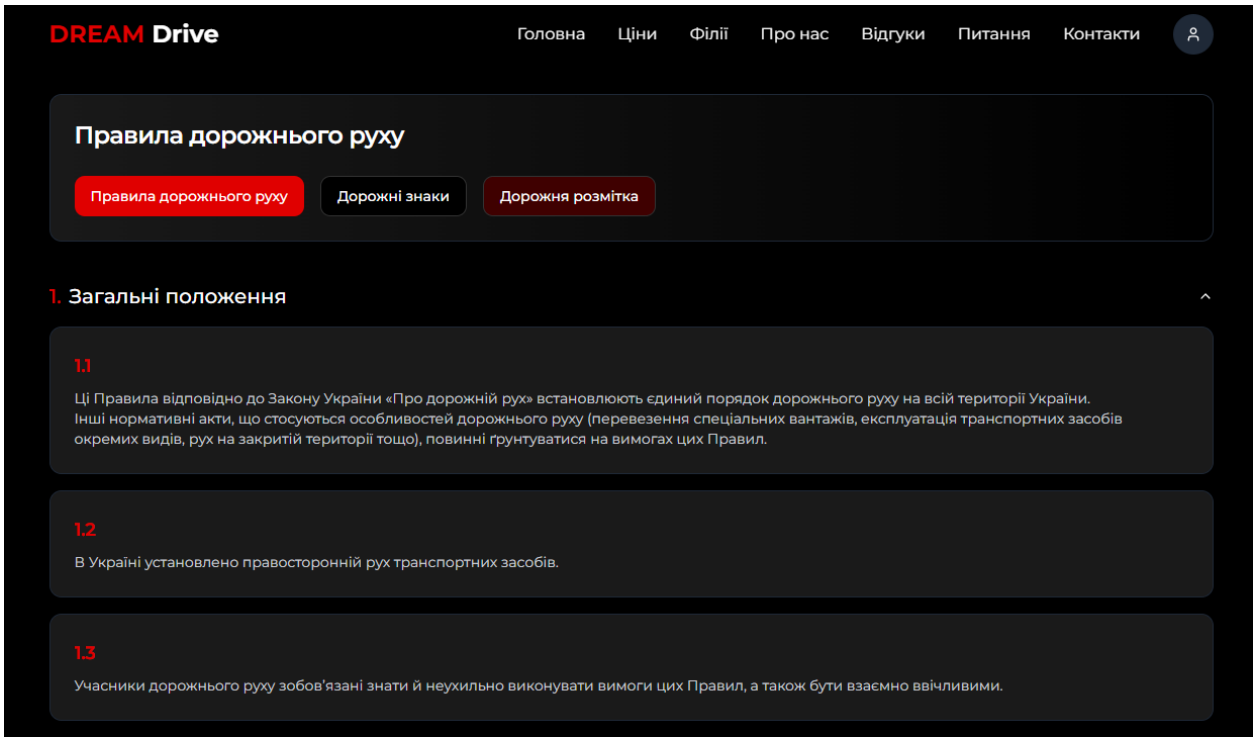


Рисунок 3.35 – Навчальні матеріали

3.3 Адміністратор

Переходимо до функціоналу адміністратора. При вході можемо зразу скористатися панеллю керування (рисунки 3.36 - 3.37).

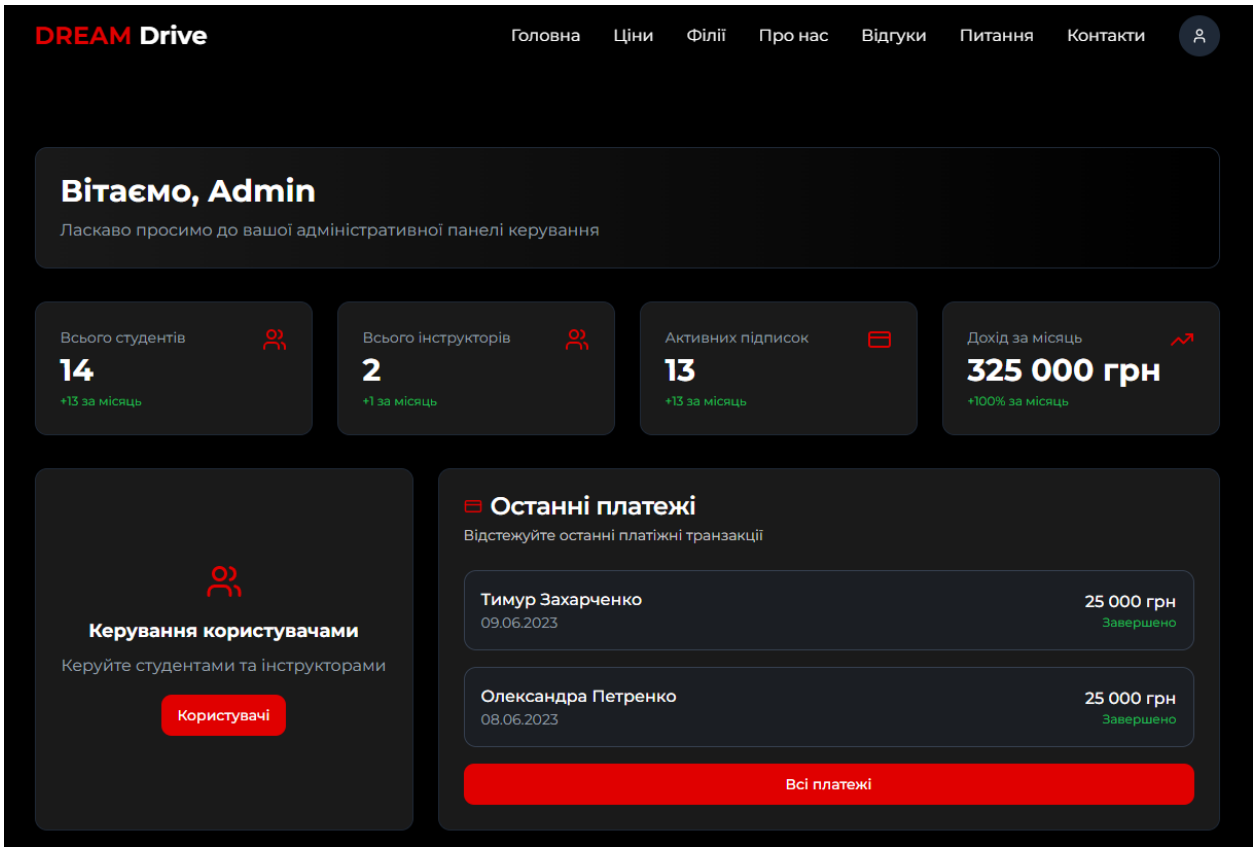


Рисунок 3.36 – Панель керування адміністратора (перша частина)

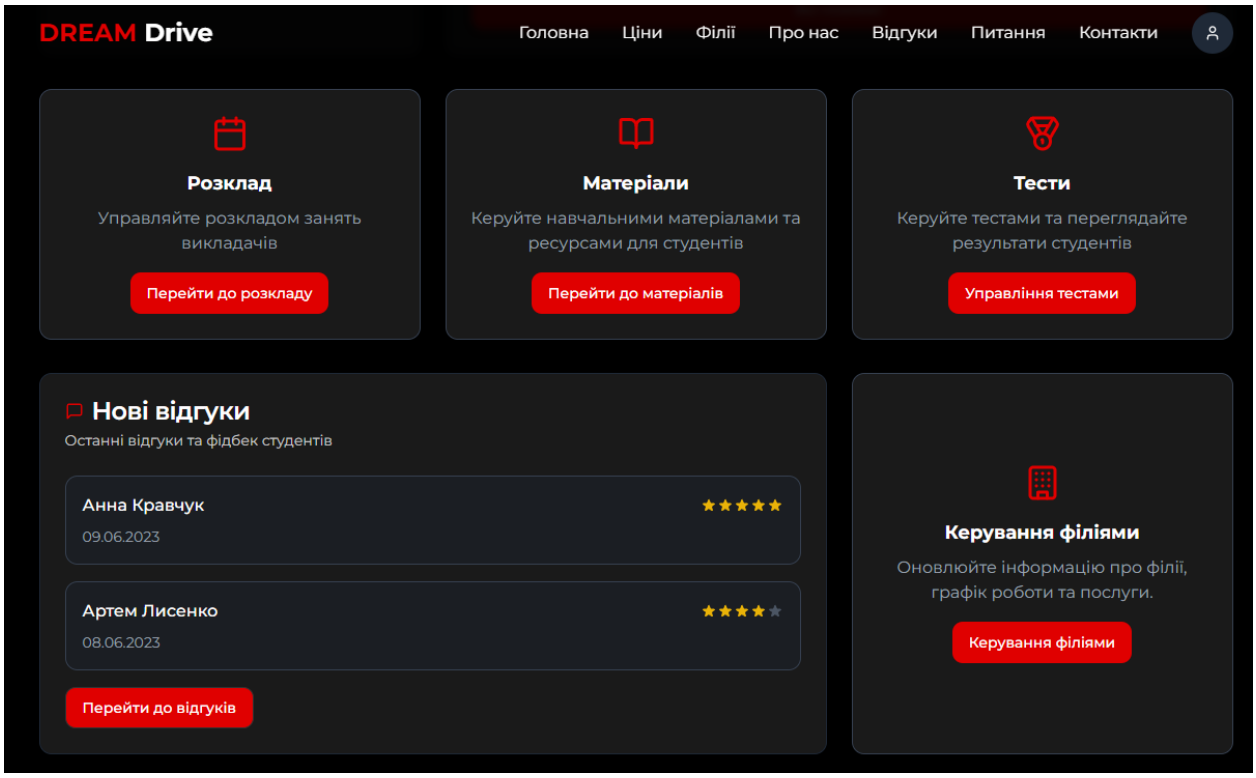


Рисунок 3.37 – Панель керування адміністратора (друга частина)

Можемо перейти до керування користувачами, де можемо додати нових користувачів або переглянути існуючих (рисунок 3.38). На сторінці платежів можемо переглянути статистику та деталі усіх платежів (рисунок 3.39).

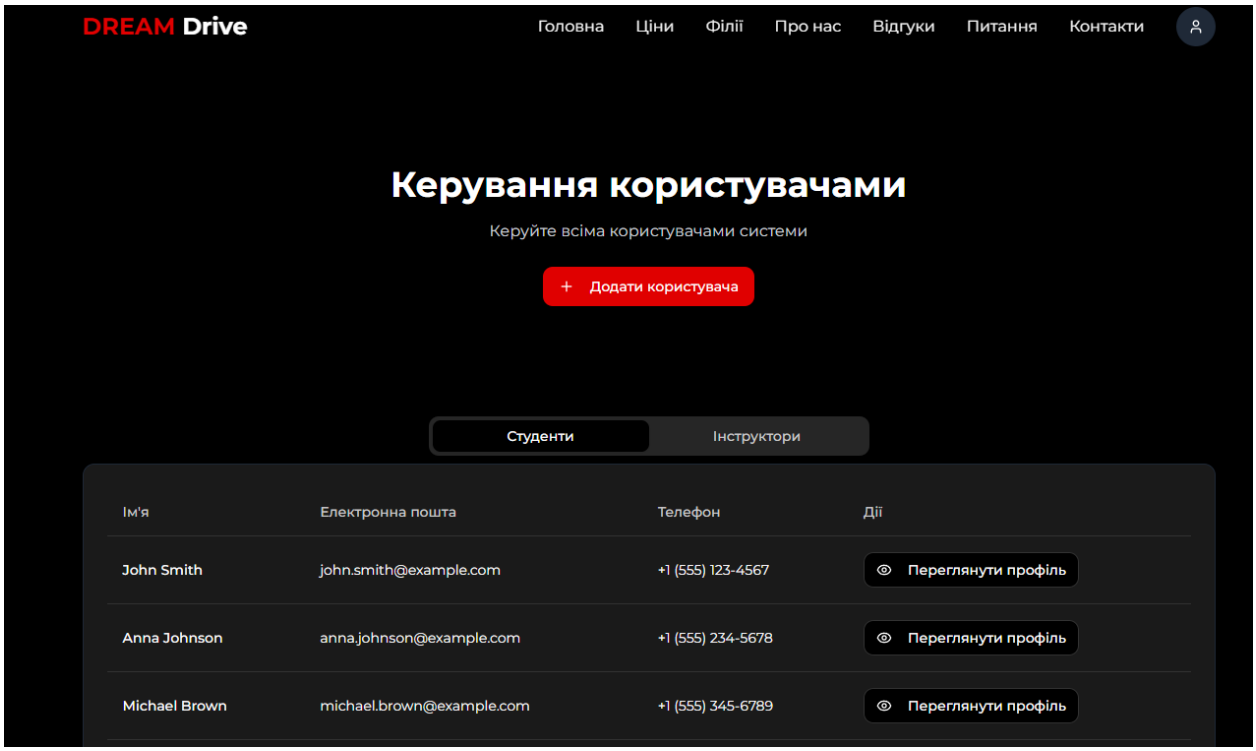


Рисунок 3.38 – Управління користувачами

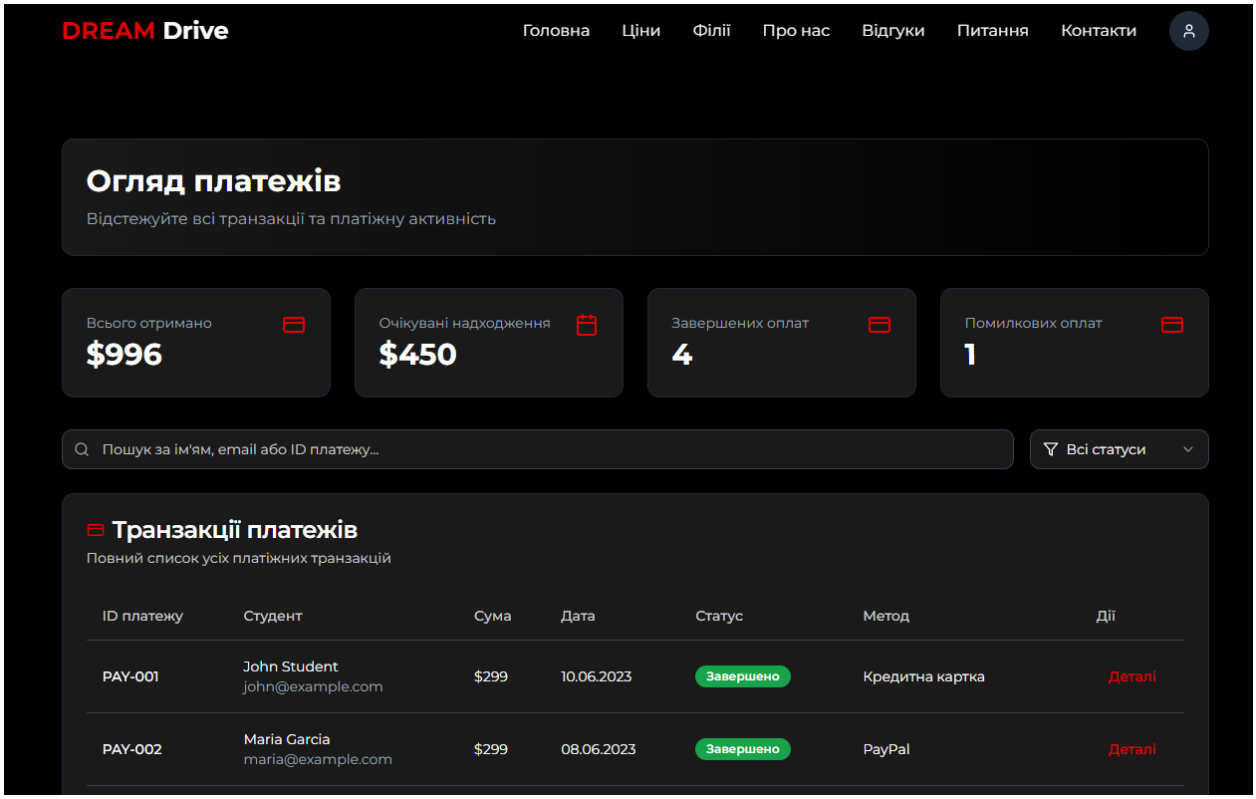


Рисунок 3.39 – Огляд платежів

Функціонал управління розкладом маємо аналогічний як у викладача, але додатково потрібно обрати викладача, розклад якого хочемо переглянути чи змінити (рисунок 3.40).

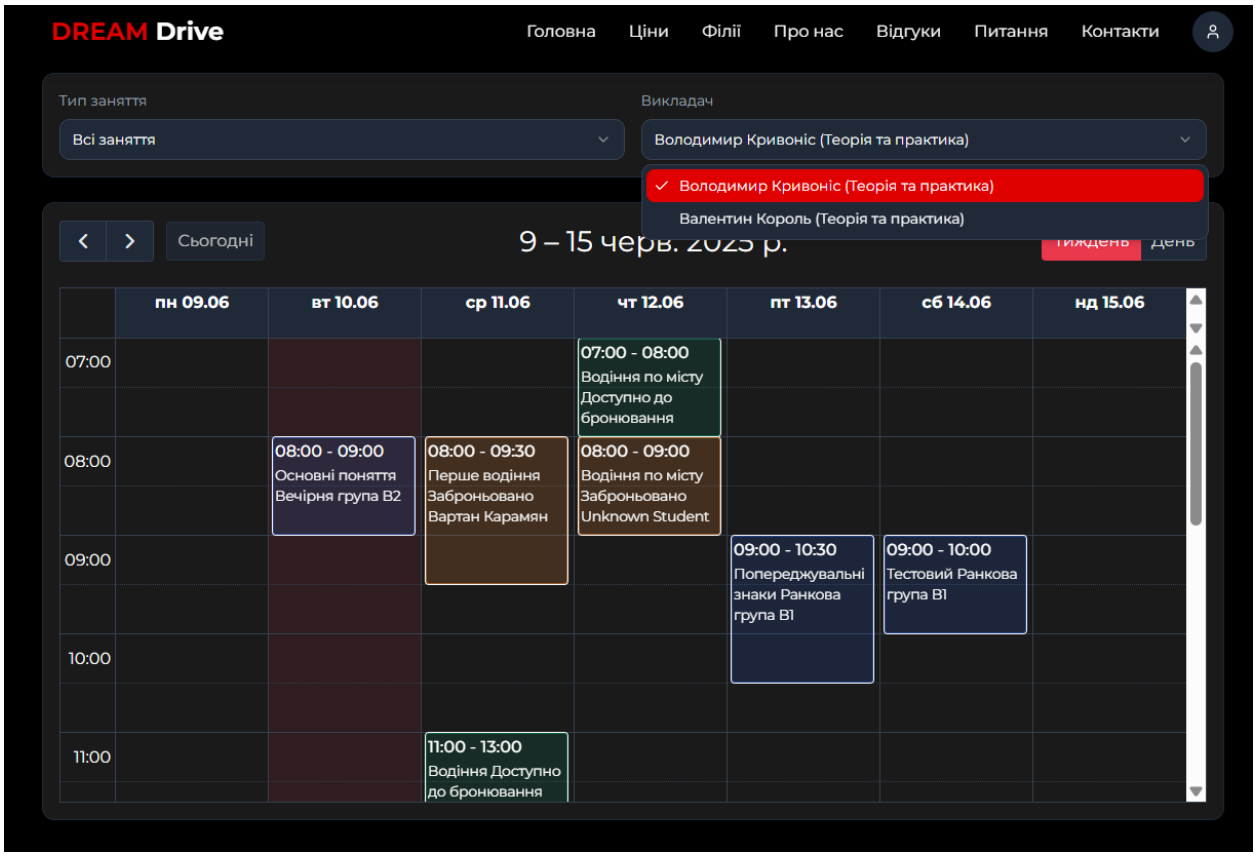


Рисунок 3.40 – Управління розкладом викладача

На сторінці з навчальними матеріалами адміністратора може додавати нові правила або редагувати і видаляти уже існуючі матеріали (рисунок 3.41).

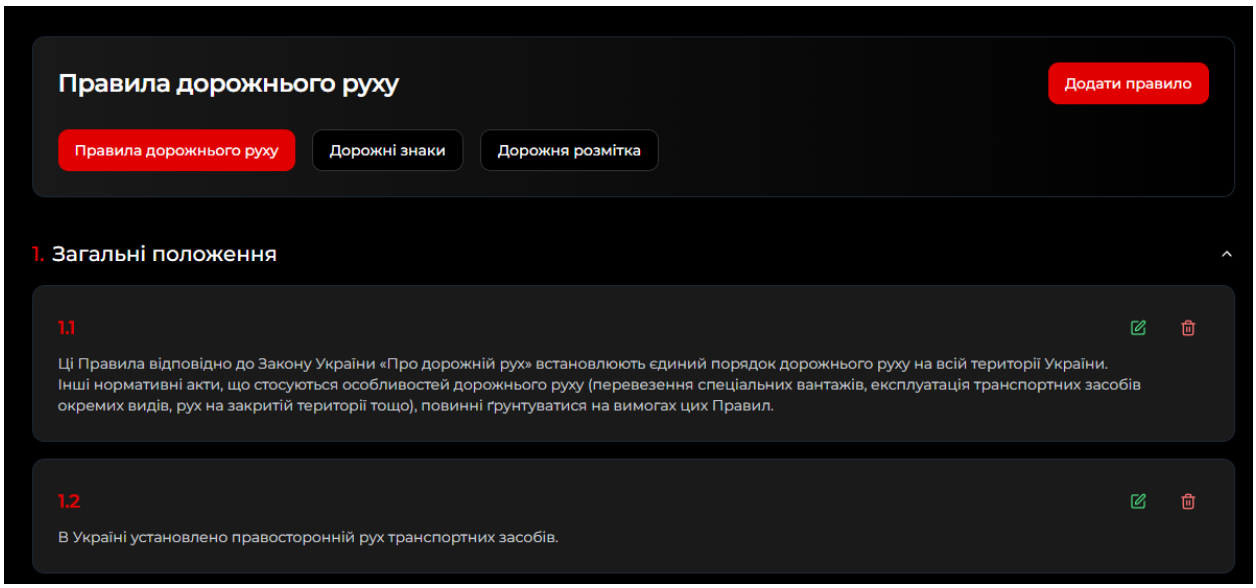


Рисунок 3.41 – Управління матеріалами

На сторінці управління контентом можемо керувати відгуками та FAQ, які відображаються на головній сторінці (рисунки 3.42 – 3.43).

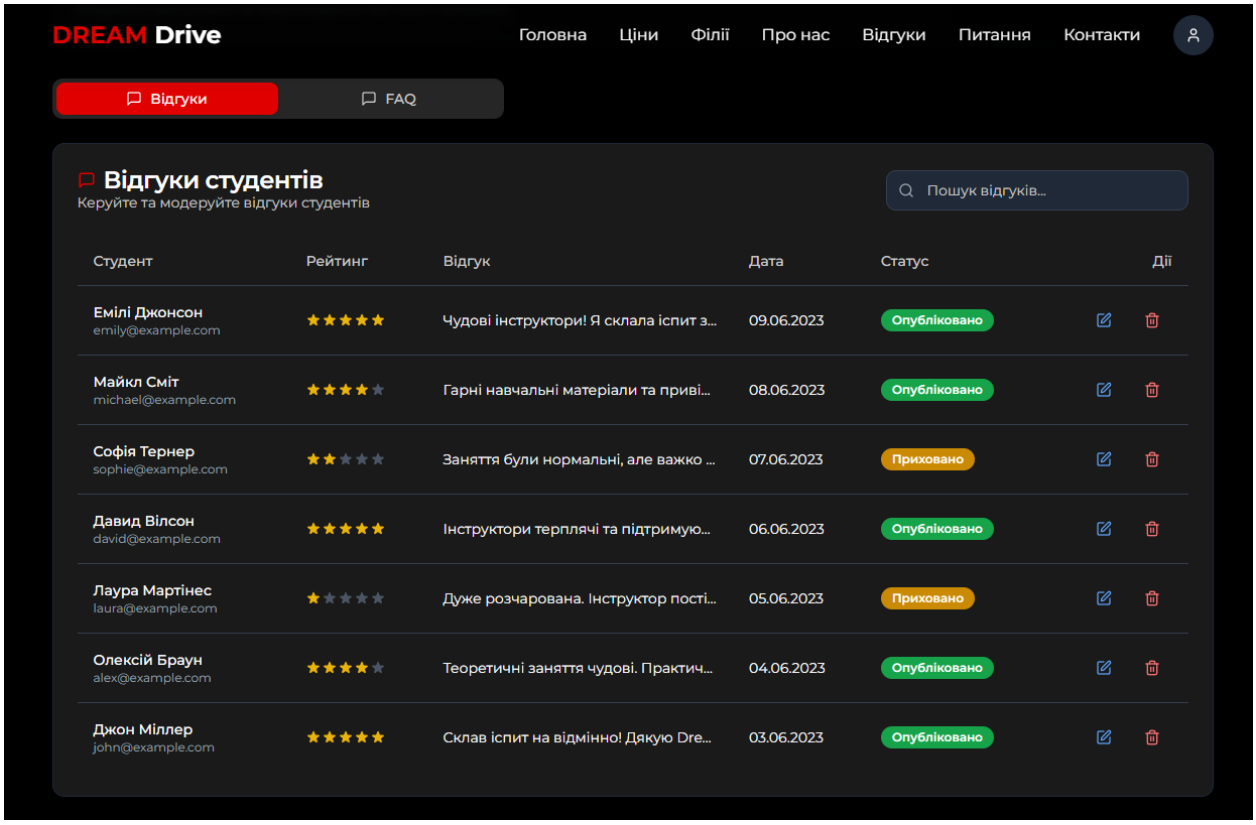


Рисунок 3.42 – Управління відгуками

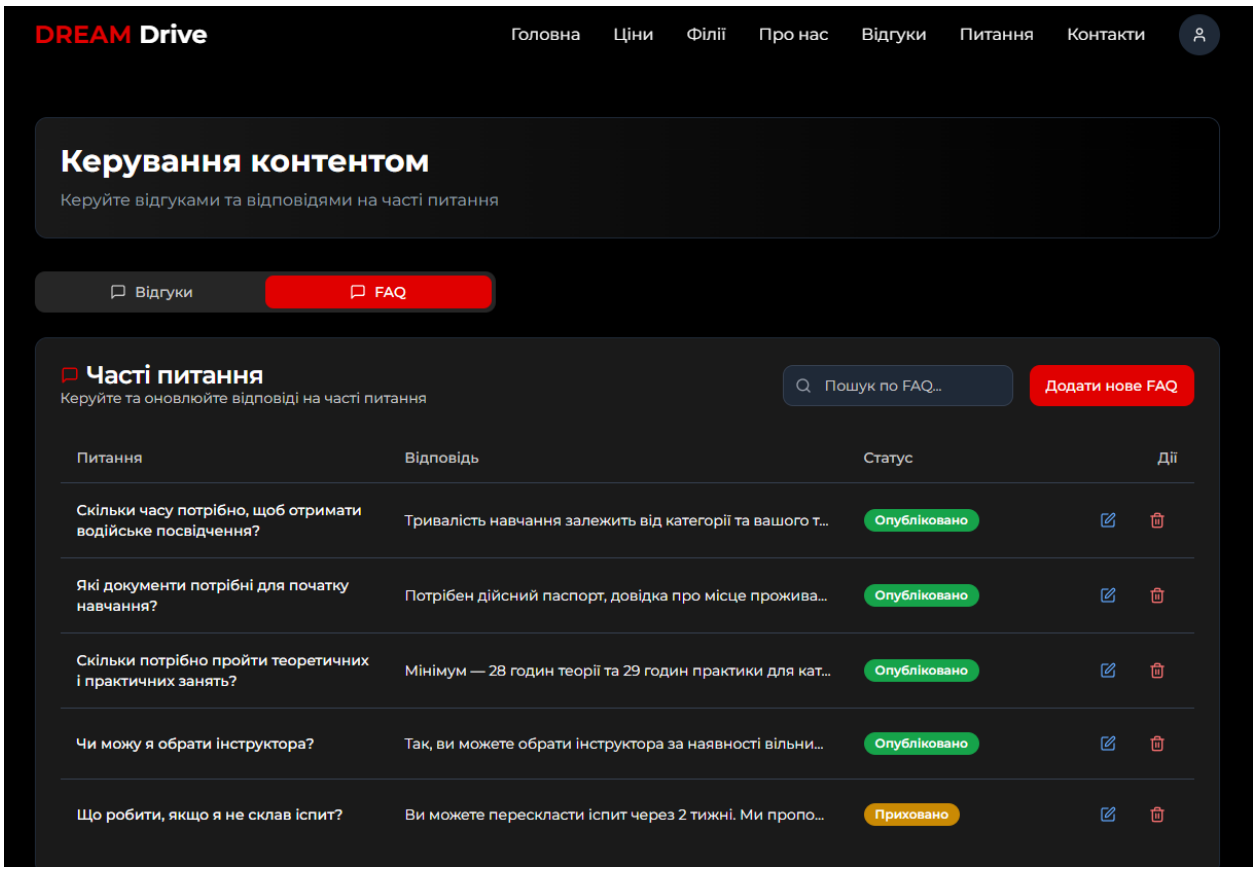


Рисунок 3.43 – Управління частими питаннями

3.4 Усі користувачі

Користувач будь-якої ролі може увійти в акаунт, щоб отримати доступ до функціоналу відповідно ролі (рисунок 3.44).

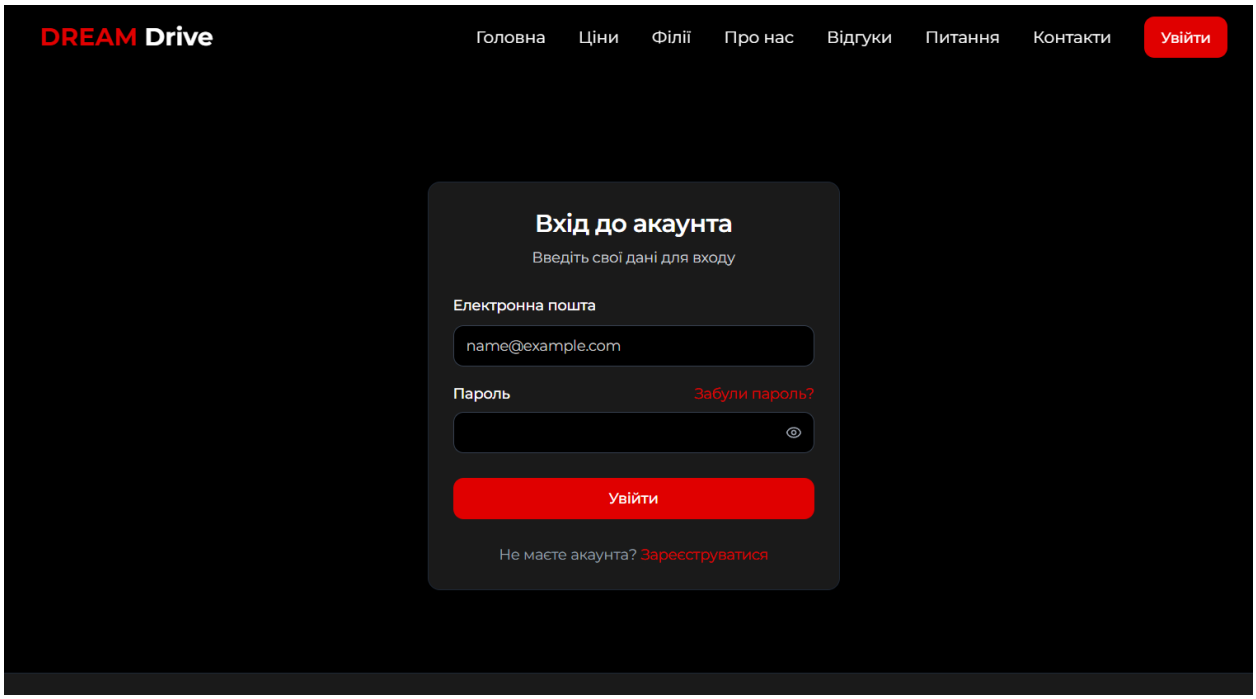


Рисунок 3.44 – Вхід до акаунту

Можемо натиснути «Забули пароль?», щоб скинути пароль за допомогою зареєстрованої пошти (рисунок 3.4). Потрібно додати свою пошту та натиснути кнопку відновлення.

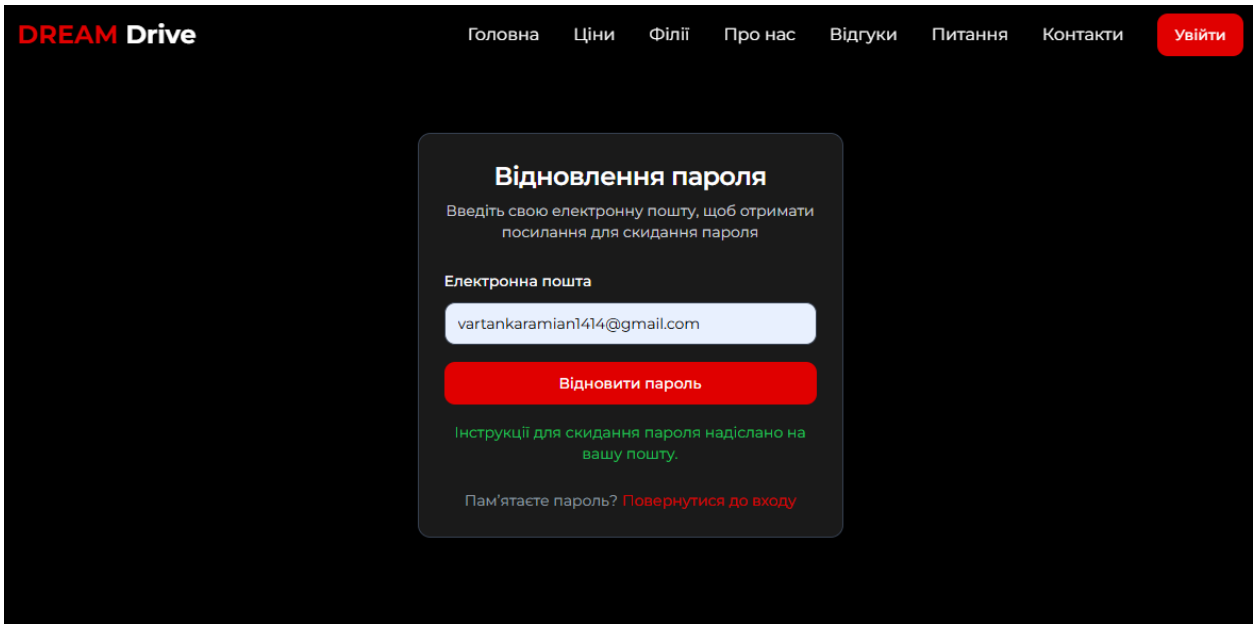


Рисунок 3.45 – Скидання паролю

Тепер перевіряємо вказану електронну пошту, куди має прийти лист з посиланням на відновлення паролю (рисунок 3.46).

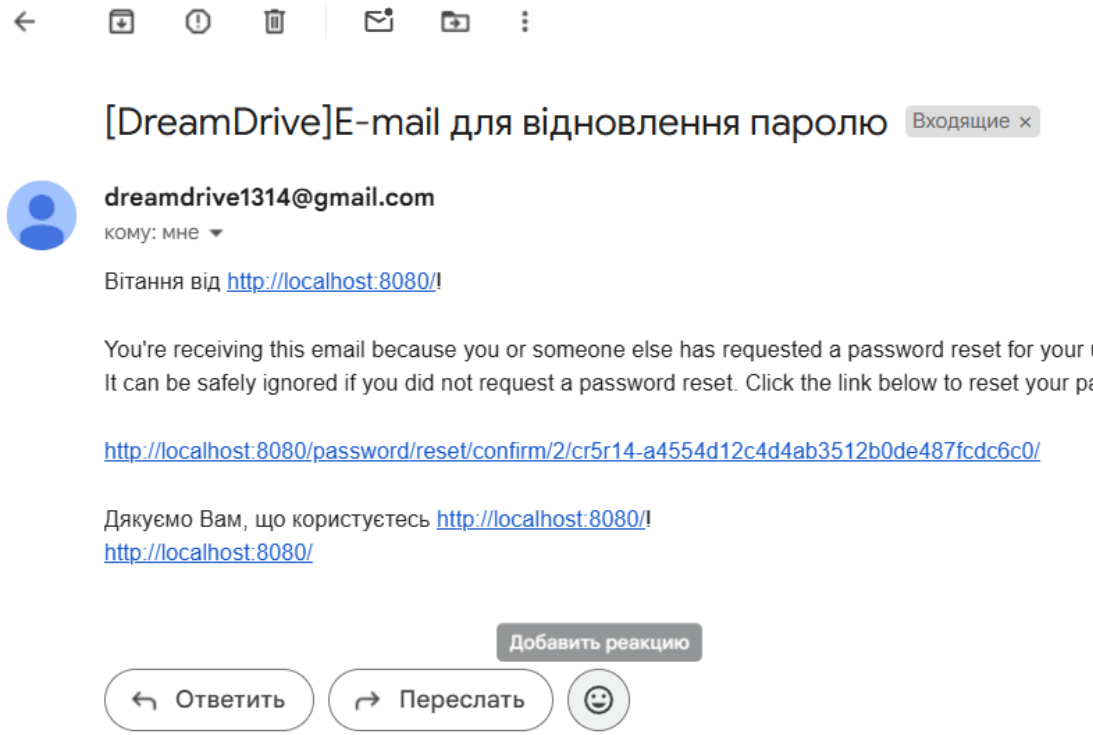


Рисунок 3.46 – Лист для відновлення паролю

Переходимо по посиланню у листі та придумуємо новий пароль (рисунок 3.47).

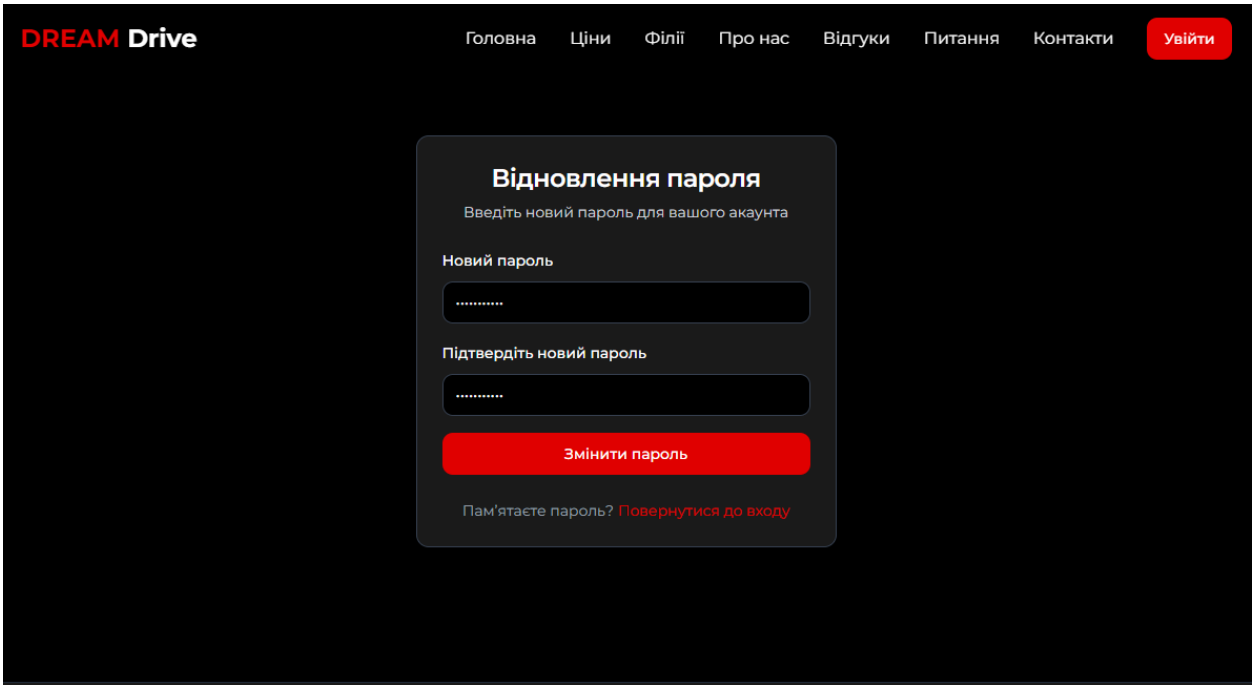


Рисунок 3.47 – Відновлення паролю

Будь-який авторизований користувач може переглянути свій профіль у вкладці «Налаштування профілю» (рисунок 3.48). Також профіль можна відредагувати, якщо натиснути відповідну кнопку, змінити необхідні поля та зберегти зміни (рисунок 3.49).

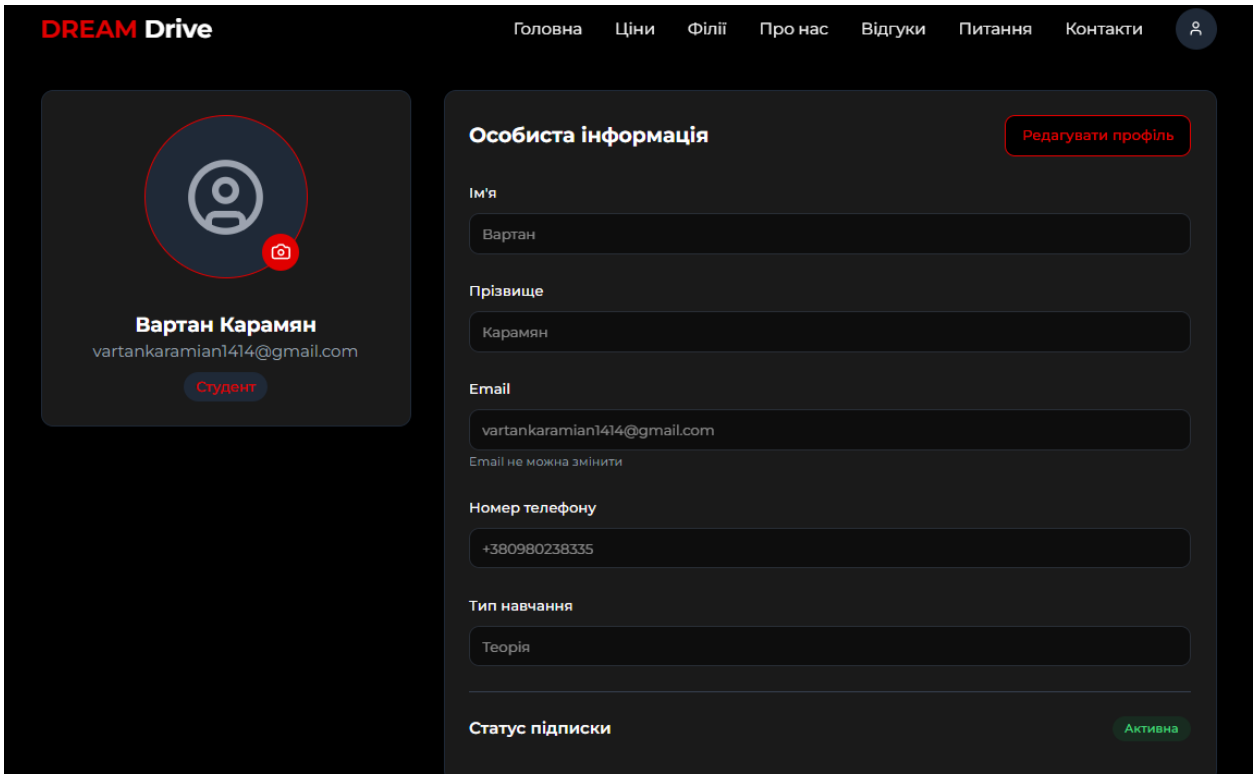


Рисунок 3.48 – Перегляд профілю

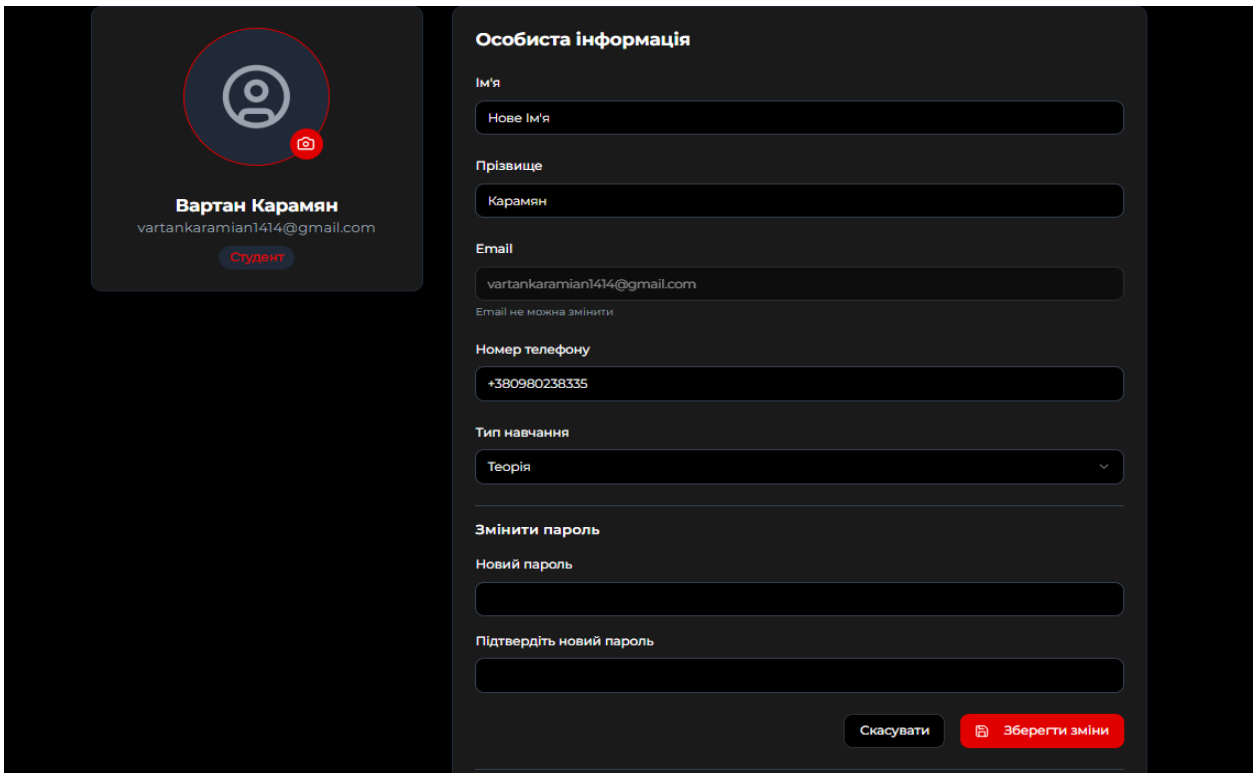


Рисунок 3.49 – Редагування профілю

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ АВТОШКОЛИ

Графічний матеріал

КПІ.ІП-1314.045440.06.99

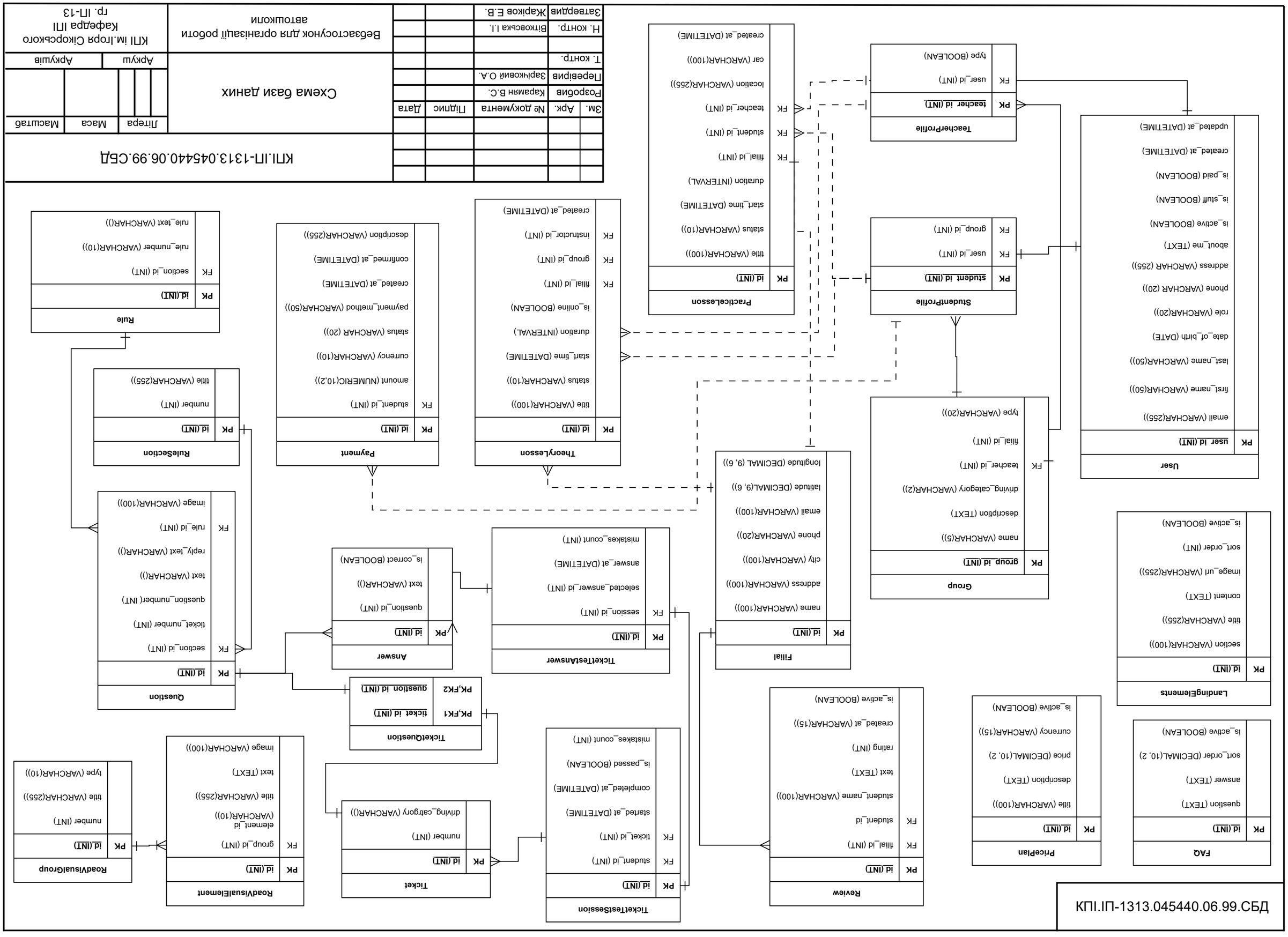
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр ЗАРІЧКОВИЙ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Вартан КАРАМЯН

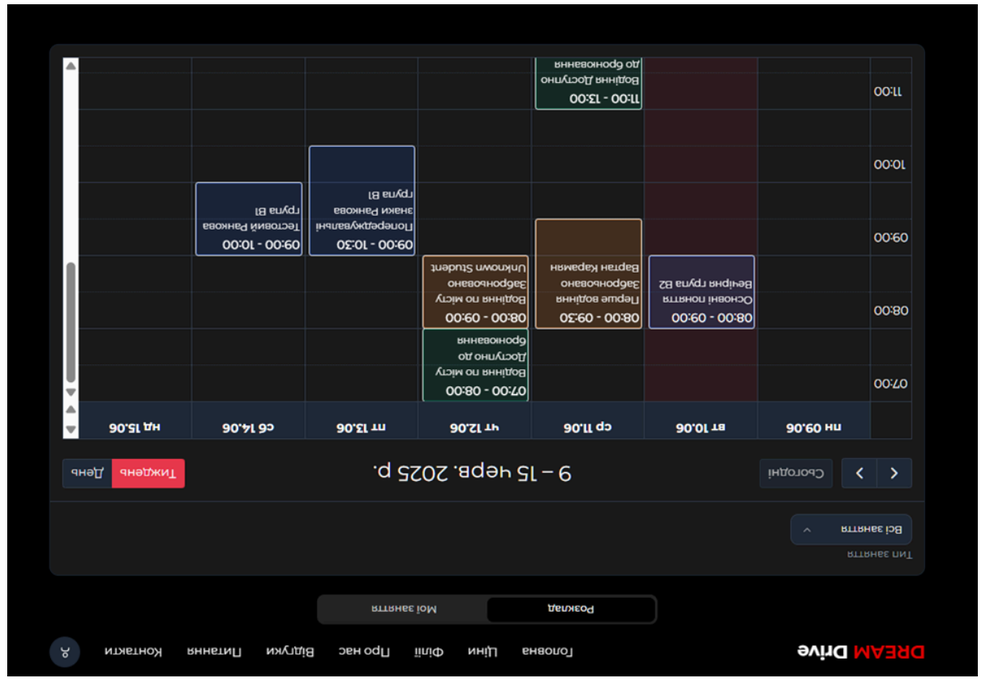
Київ – 2025

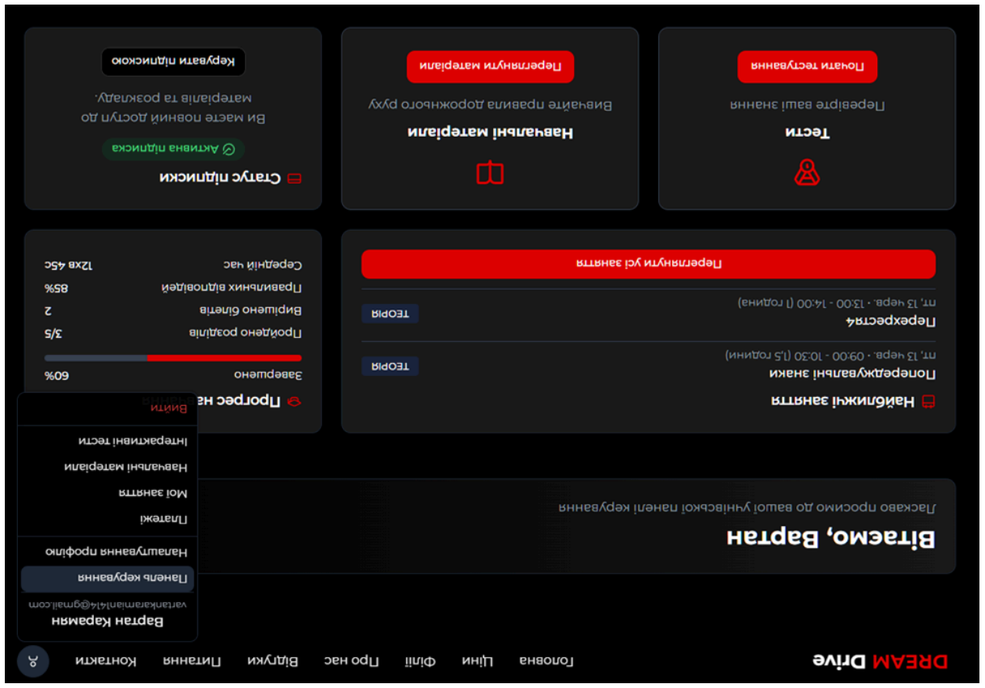
						КПІ.ІП-1313.045440.06.99.ССВ	
Зм.	Арк.	№ докум.	Підп.	Дата			
Розробив		Карамян В.С.				Схема структура варіантів використань	
Перевірив		Зарічковий О.А.					
Т. контр.							
Н. контр.		Вітковська І.І.					
Затвердив		Жарікова Е.В.					
						Літ.	Маса
						Аркуш	Аркушів
						Ведомість для організації робот автошколи	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-13

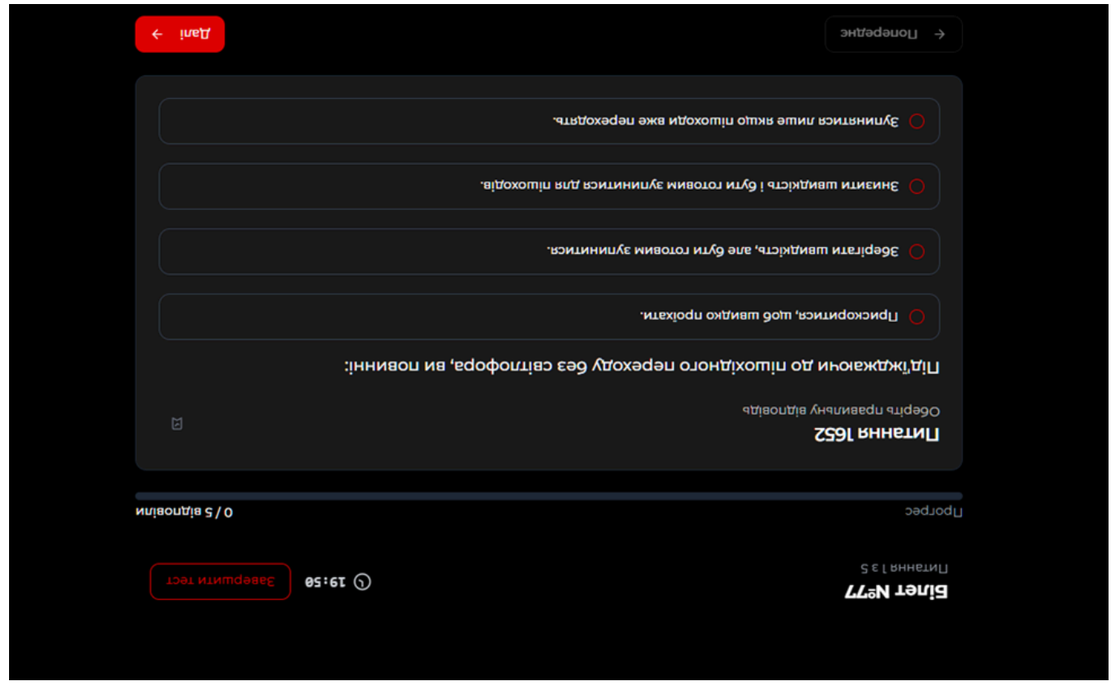


КПІ.ІП-1313.045440.06.99.СБД

								КПІ.ІП-1313.045440.06.99.СБД	
Змн.	Арк.	№ докумен.	Підпис	Дата					
Розробив		Карамич В.С.						Схема бази даних	
Перевірив		Зарічковий О.А.							
Т. контр.								Арк.	Аркушів
Н. контр.		Бітковська І.І.							
Затвердив		Жарікова Е.В.						Веб-застосунок для організації роботи автошколи	
								КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-13	
Літера		Маса		Масштаб					







КПІ.ІП-1313.045440.06.99.КЕ

							КПІ.ІП-1313.045440.06.99.КЕ	
3м.	Арк.	№ докумeнта	Підпис	Дата	Кресленна вигляду екранних форм		Літера	Маса	Масштаб
Розробив		Карамян В.С.							
Перевірив		Зарічковий О.А.							
Т. контр.							Аркуш		Аркушів
Н. контр.		Бітковська І.І.			Вебзастосунок для організації роботи автошколи		КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-13		
Затвердив		Жарікова Е.В.							