

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи освітнього ступеня «бакалавр» за
спеціальністю 121 «Інженерія програмного забезпечення» (освітня програма
«Інженерія програмного забезпечення»)

на тему:

«Вебплатформа онлайн-школи»

Виконав студент групи ІПЗ-22-2
ЗАРЕМБИЦЬКИЙ Єгор Юрійович

Керівник роботи:
ЧИЖМОТРА Олексій Володимирович

Рецензент:
ФАНТ Микола Олександрович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк

«27» лютого 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу

Здобувач вищої освіти: **ЗАРЕМБИЦЬКИЙ Єгор Юрійович**

Керівник роботи: **ЧИЖМОТРЕ Олексій Володимирович**

Тема роботи: **«Вебплатформа онлайн-школи»**

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: об'єктом дослідження є вебплатформа для організації навчання онлайн-школи. Предметом дослідження є підходи та засоби розробки сучасного веб-застосунку, що забезпечує модерацію користувачів, взаємодію вчителів та учнів, організацію уроків.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	01.03.2026	01.03.2026
2	Чижмотря О.В.	01.03.2026	01.03.2026
3	Чижмотря О.В.	01.03.2026	01.03.2026

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебплатформи онлайн-школи та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить текстову частину, викладену на 114 сторінках друкованого тексту. Список використаних джерел містить 30 найменувань і займає 3 сторінки. У роботі наведено 61 ілюстрацію та 16 таблиць. Також у роботі є 7 сторінок додатків. Загальний обсяг роботи – 114 сторінок.

Головною метою є створення сучасної та зручної вебплатформи для управління навчальним процесом онлайн-школи, яка забезпечує можливість роботи вчителів, надає змогу учням вільно обирати предмет та вчителя, дозволяє адміністрації школи виконувати процес модерації користувачів, а також вести аудит оплат, проведених під час роботи додатку.

У процесі роботи було проаналізовано напрями використання інформаційних технологій, сформовано постановку задачі та проведено аналіз існуючих порталів онлайн-шкіл. Результатом є вибір архітектури вебплатформи, котра задовільнить поставлені вимоги. Були визначені варіанти використання системи та сформовано її об'єктно-орієнтовану структуру. Було продумано та розроблено структуру бази даних, реалізовано алгоритми роботи, враховуючи інтерфейс та налаштування взаємодії різних компонентів додатку між собою. Було проведено детальне тестування програмного продукту задля забезпечення стабільної роботи та безпеки користувачів, що підтверджується результатом наведеним в третьому розділі.

КЛЮЧОВІ СЛОВА: REACT, NESTJS, CORS, TYPESCRIPT, JAVASCRIPT, JWT-АВТОРИЗАЦІЯ, TYPEORM, POSTGRESSQL, BASE64, REST API, УРОК, АККАУНТ, ПРЕДМЕТ, ВИКЛАДАЧ.

					ІПЗ.КР.Б-121-26-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Зарембицький Є.Ю.			Вебплатформа онлайн-школи	Літ.	Арк.	Аркуші
Керівник		Чижмоторя О.В.					3	114
Рецензент		Фант М.О.			Пояснювальна записка	Житомирська політехніка група ІПЗ-22-2		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification work consists of an online school web platform and an explanatory note. The explanatory note to the qualification work contains a textual part presented on 114 pages of printed text. The list of references includes 30 sources and occupies 3 pages. The work contains 61 illustrations and 16 tables. The work also includes 7 pages of appendices. The total volume of the work is 114 pages.

The main objective is to create a modern and user-friendly web platform for managing the educational process of an online school, which provides the ability for teachers to work, allows students to freely choose a subject and a teacher, enables the school administration to moderate users, and also to conduct an audit of payments made during the operation of the application.

During the work process, the directions of information technology usage were analyzed, the problem statement was formulated, and an analysis of existing online school portals was conducted. The result is the selection of a web platform architecture that satisfies the defined requirements. System use cases were identified, and its object-oriented structure was formed. The database structure was designed and developed, and the operational algorithms were implemented, taking into account the interface and configuration of interaction between various components of the application. Detailed testing of the software product was carried out to ensure stable operation and user security, which is confirmed by the results presented in the third section.

KEYWORDS: REACT, NESTJS, CORS, TYPESCRIPT, JAVASCRIPT, JWT AUTHENTICATION, TYPEORM, POSTGRESQL, BASE64, REST API, LESSON, ACCOUNT, SUBJECT, TEACHER.

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	10
1.1. Аналіз аналогів програмного продукту.....	10
1.2. Постановка задачі	17
1.3. Вибір архітектури веборієнтованої системи.....	20
Висновки до першого розділу	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ.....	24
2.1. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	24
2.2. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	27
2.3. Розробка бази даних системи	37
2.4. Проєктування та реалізація алгоритмів роботи системи.....	49
2.5. Реалізація функціоналу системи	53
Висновки до другого розділу.....	60
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СЕРВІСОМ	62
3.1. Порядок встановлення та налаштування параметрів системи.....	62
3.2. Структура інтерфейсу вебзастосунку	65
3.3. Тестування роботи програмного забезпечення	95
Висновки до третього розділу	100
ВИСНОВКИ.....	102
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	104
ДОДАТКИ.....	107

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

ПЗ – програмне забезпечення

CORS – Cross-Origin Resource Sharing (спільне використання ресурсів з різних джерел)

ORM – Object-Relational Mapping (Об’єктно-реляційне відображення)

Rest api – Representational State Transfer Application Programming Interface (Інтерфейс прикладного програмування на основі передачі репрезентативного стану)

JWT – JSON Web Token (Вебтокен JSON)

HTTP – Hypertext Transfer Protocol (Протокол передавання гіпертексту)

HTML – Hypertext Markup Language (Мова розмітки гіпертексту)

JSON – JavaScript Object Notation (Нотація об’єктів JavaScript)

CSS – Cascading Style Sheets (Каскадні таблиці стилів)

UI – User interface (Користувацький інтерфейс)

UX – User experience (Користувацький досвід)

CRUD – Create, Read, Update Delete (Створити, зчитати, оновити, видалити), стандартний набір операцій з даними в БД

SQL - Structured Query Language (структурована мова запитів)

DTO – Data transfer object (об’єкт передавання даних)

Env - Environment Variables (змінні середовища)

API - Application Programming Interface (Інтерфейс прикладного програмування)

ВСТУП

Актуальність теми зумовлена стрімким розвитком та інтеграцією дистанційного навчання в освітній процес. Пандемія та тривалий воєнний стан в Україні висунули нові вимоги до безпеки, гнучкості та доступності освіти. Наявні великі платформи часто є перевантаженими або не забезпечують прямої, безпечної та прозорої взаємодії між репетиторами, учнями та адміністрацією. Створення спеціалізованих вебплатформ онлайн-шкіл, які автоматизують пошук викладачів, процес запису на заняття, забезпечують модерацію контенту та прозору систему відгуків, є важливим кроком для підвищення якості надання освітніх послуг та безпеки користувачів.

Розвиток інтернет-технологій докорінно змінив відношення до навчання з того часу. Завдяки йому діти можуть безпечно відвідувати заняття під час воєнного стану чи негоди. Раніше при таких умовах оголошувався карантин, навчання зупинялось на невизначений термін. Створені тоді портали та сервіси не зупинили свого розвитку і допомагають здобувачам освіти і сьогодні. На допомогу прийшли портали приватних онлайн-шкіл, котрі реалізовували зв'язок між викладачами та учнями. Вони дозволяють учням обирати потрібний їм предмет та вчителя, дають змогу учням напряду взаємодіяти з адміністрацією шкіл, автоматизують процес запису на курси та спрощують процес пошуку репетиторів.

Метою випускної кваліфікаційної роботи є проєктування та програмна реалізація вебплатформи онлайн-школи репетиторства з автоматизованими модулями підбору викладачів, модерації взаємодії користувачів та рейтингової оцінки.

Практичними завданнями кваліфікаційної роботи є

- аналіз практичних і теоретичних вимог до вебплатформ онлайн-шкіл орієнтованих на репетиторство;
- визначення структури та архітектури системи відповідно до стандартів сьогодення;

- оцінка вебплатформ-аналогів, аналіз їх позитивних та негативних аспектів;
- вибір та обґрунтування технологій, а також інструментарію, котрий буде використано у процесі розробки;
- розробка ключових аспектів системи: сервеної та клієнтської частини, взаємодії між ними;
- формування кінцевого програмного продукту та його тестування, задля досягнення оптимального результату та надійного функціонування надалі;

Об'єктом дослідження є процес автоматизації організації навчального процесу та взаємодії між учасниками в онлайн-школі репетиторства.

Предметом дослідження є сучасні вебтехнології, архітектурні рішення та програмні засоби розробки відмовостійкої вебплатформи онлайн-школи репетиторства.

В процесі розробки використовуються інструменти взаємодії компонентів клієнтської та серверної частини разом із зв'язком до БД з можливістю безпечного зв'язку користувачів з цими ресурсами без ризику втрати особистих даних, використано модульну структуру, об'єктно-орієнтований підхід. Всі ці показники забезпечуються використанням сучасного стеку веб-технологій, що надає йому більшу продуктивність, меншу завантаженість клієнтської частини та дає підтримку системи актуальних API.

В роботі використано такі методи системного та порівняльного аналізу, методи об'єктно-орієнтованого проєктування, сучасні технології вебпрограмування, проєктування реляційних/нереляційних баз даних та методи тестування програмного забезпечення.

Практичним значенням отриманих результатів є розроблена вебплатформа є повноцінним програмним рішенням, готовим до використання у сфері надання освітніх послуг. Практична цінність роботи полягає в можливості безпосереднього впровадження створеного продукту в діяльність існуючих або новостворених онлайн-шкіл, репетиторських центрів та приватних викладачів.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						8

Використання платформи дозволить реальним освітнім установам автоматизувати процеси пошуку викладачів, організації запису на заняття та загального адміністрування навчального процесу.

Апробація. За темою кваліфікаційної роботи було опубліковано 3 тези конференцій [1, 2, 3].

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Аналіз аналогів програмного продукту

Перед початком планування структури проекту та початком його розробки потрібно оцінити аналогічний програмний продукт, котрий вже наявний на ринку. Ця дія допоможе зрозуміти які рішення вже використовуються конкурентами також це надасть можливість оцінити їхні вдалі якості та вивчити помилки інших не повторюючи їх. Це дозволить зрозуміти наявні тренди в галузі, її стан в цілому та надасть розуміння в якому напрямку краще розвивати проєкт.

Протягом аналізу сайтів-аналогів онлайн-шкіл орієнтованих на репетиторство було оцінено такі три типові сайти: BUKI, Mathema, TeMA. Нижче наведено детальний огляд кожної платформи.

Платформа BUKI

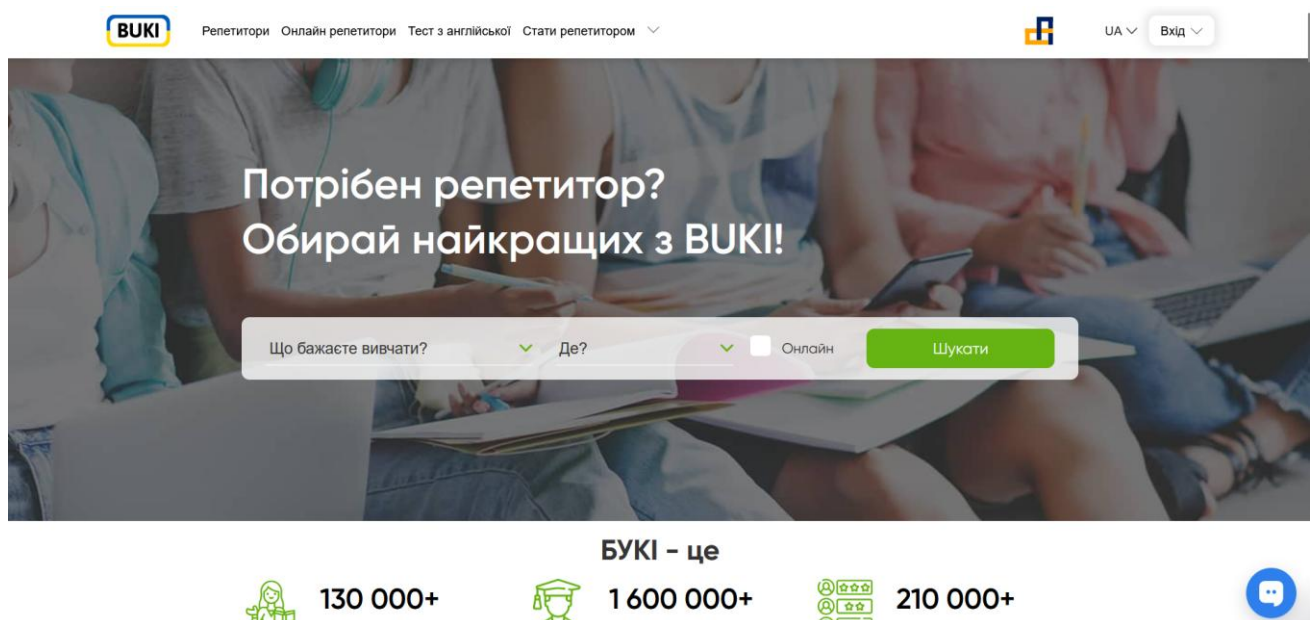


Рис. 1.1 – Вигляд головної системи платформи BUKI

BUKI є масштабною українською вебплатформою для організації індивідуального онлайн-навчання у форматі репетиторства (рис. 1.1), яка реалізує архітектуру маркетплейсу викладачів. З технічної точки зору платформа надає повний набір функціональних модулів для взаємодії трьох основних ролей: учня, викладача та адміністратора. Система включає механізми реєстрації та

авторизації користувачів, створення та керування профілями викладачів, каталог предметів і напрямків навчання, а також розширену систему пошуку з фільтрацією за ціною, рейтингом, досвідом та форматом занять. Важливою складовою є система рейтингів і відгуків, яка формує довіру між користувачами та впливає на ранжування викладачів у результатах пошуку. Платформа підтримує асинхронну комунікацію між учнем і репетитором через заявки та повідомлення, а також дозволяє узгоджувати формат занять без жорсткої прив'язки до внутрішнього відеосервісу. Окремий технічний напрямок розвитку - BUKI School, де реалізована централізована модель навчання з внутрішнім контролем якості викладачів, обліком занять, розкладом та супроводом учнів. Загалом BUKI демонструє приклад великої, функціонально насиченої веб-платформи, орієнтованої на масштабування, роботу з великим обсягом користувачів і складну бізнес-логіку взаємодії між сторонами навчального процесу. Порівняльну характеристику наведено в таблиці нижче (табл. 1.1).

Таблиця 1.1

Порівняльна характеристика платформи BUKI

Переваги та сильні сторони	Недоліки та обмеження
Реалізована багаторольова система користувачів	Відсутність єдиного інтегрованого середовища проведення онлайн уроків;
Розвинена система профілів з великою кількістю параметрів і метаданих	Слабка інтеграція навчального процесу безпосередньо в платформу
Потужний модуль пошуку та фільтрації за декількома критеріями	Мінімальна автоматизація навчальних сценаріїв
Система рейтингів і відгуків як окремий функціональний підмодуль;	Платформа виступає більше як агрегатор, ніж як повноцінна LMS-система;

Підтримка масштабно бази користувачів і контенту;	Обмежені інструменти аналітики для учнів і викладачів. як оце бажано було б перевести в табличку і назву таблички треба
Модуль заявок і повідомлень для комунікації між користувачами;	
Можливість розширення функціоналу;	
Архітектура, орієнтована на маркетплейс-модель і подальше масштабування.	

Сервіс Mathema [2]

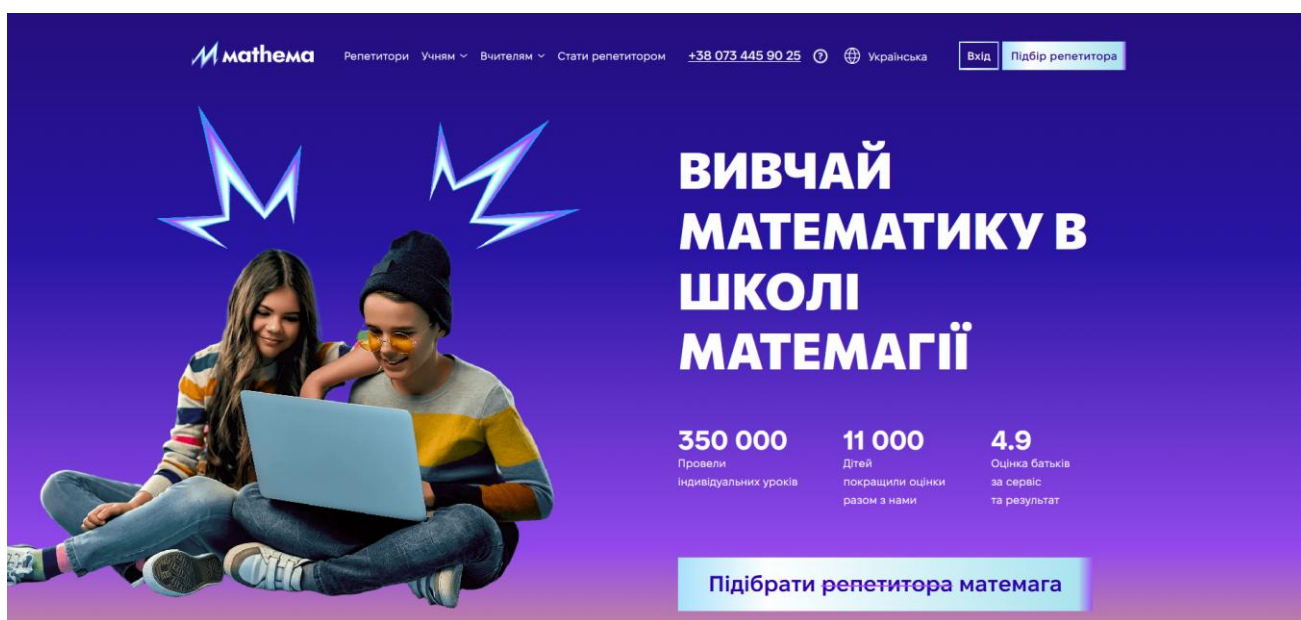


Рис. 1.2 – Вигляд головної системи платформи Mathema

Mathema - це українська вебплатформа онлайн-навчання, орієнтована на індивідуальні заняття формату 1 на 1, яка реалізує модель централізованої онлайн-школи з керованим навчальним процесом (рис. 1.2). На відміну від класичного маркетплейсу репетиторів, Mathema використовує більш закриту архітектуру, де платформа не лише з'єднує учня з викладачем, а й безпосередньо керує логікою навчання. З технічної точки зору сервіс включає модулі обліку

учнів, викладачів та занять, систему розкладів, абонементів і контролю прогресу. Користувач взаємодіє не з каталогом викладачів, а з навчальною системою, яка автоматично підбирає репетитора відповідно до рівня знань, цілей і віку учня. Платформа підтримує повністю онлайн-формат проведення занять, включно з відеозв'язком, навчальними матеріалами та внутрішніми інструментами взаємодії. Значна частина логіки зосереджена на автоматизації навчального процесу: фіксації відвідувань, обліку кількості проведених занять, управлінні навчальними пакетами та аналітиці успішності. Таким чином, Mathema є прикладом вебплатформи, яка поєднує функції сервісу онлайн-репетиторства та елементів LMS-системи з чітко визначеним навчальним сценарієм. Порівняльну характеристику наведено в таблиці нижче (табл. 1.2).

Таблиця 1.2

Порівняльна характеристика платформи Mathema

Переваги та сильні сторони	Недоліки та обмеження
Централізована архітектура онлайн-школи з керуванням навчальним процесом	Відсутність відкритого маркетплейсу викладачів
Автоматизований підбір викладача без участі учня	Обмежена свобода вибору репетитора для користувача
Вбудований облік занять, відвідувань та навчальних пакетів	Складніша архітектура порівняно з простими агрегаторами
Система розкладів і планування уроків	Менша гнучкість навчального процесу для нестандартних сценаріїв
Інтеграція онлайн-занять без потреби сторонніх сервісів	Висока залежність користувача від внутрішньої логіки платформи
Наявність внутрішніх навчальних матеріалів	

Базова	аналітика	навчального
прогресу учнів		

Платформа ТеМа

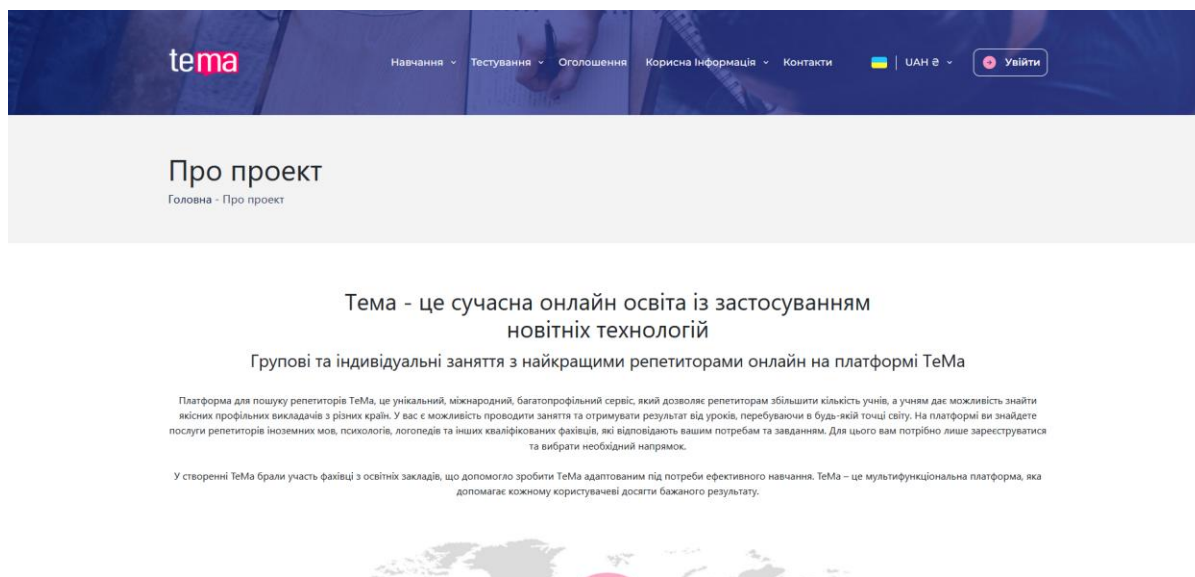


Рис. 1.3 – Вигляд головної системи платформи ТеМа

ТеМа - це українська вебплатформа для пошуку та організації індивідуальних онлайн-занять з репетиторами (рис. 1.3), яка поєднує риси класичного маркетплейсу викладачів і базові інструменти керування навчальним процесом. З технічної точки зору платформа реалізує модель каталогу викладачів з особистими профілями, системою пошуку та фільтрації, а також механізмами комунікації між учнем і викладачем. Користувач має можливість самостійно обирати репетитора за предметом, рівнем підготовки, вартістю занять та іншими параметрами, що свідчить про орієнтацію сервісу на індивідуальне навчання формату 1 на 1. Платформа підтримує онлайн-формат взаємодії, проте більшість навчального процесу відбувається поза межами єдиного інтегрованого середовища, що характерно для агрегаторів репетиторських послуг. Архітектура ТеМа зосереджена на забезпеченні базових функцій: управління обліковими записами, відображення профілів викладачів, обробки заявок на заняття та первинної координації між сторонами навчального процесу. Таким чином, ТеМа є прикладом відносно легкої за складністю вебплатформи, орієнтованої на швидкий пошук викладача без глибокої автоматизації навчальних сценаріїв.

Порівняльну характеристику наведено в таблиці нижче (табл. 1.3).

Таблиця 1.3

Порівняльна характеристика платформи ТеМа

Переваги та сильні сторони	Недоліки та обмеження
реалізація маркетплейс-архітектури з каталогом викладачів	відсутність повноцінної LMS-системи
система профілів репетиторів з базовими характеристиками	мінімальна автоматизація навчального процесу
пошук і фільтрація за предметами та параметрами занять	відсутність вбудованих інструментів для проведення онлайн-уроків
підтримка індивідуального формату навчання 1:1	слабка аналітика прогресу учнів
проста і зрозуміла логіка взаємодії користувачів	слабка аналітика прогресу учнів
менша складність архітектури порівняно з повноцінними LMS	обмежені можливості контролю якості навчання
можливість швидкого масштабування каталогу викладачів	залежність від сторонніх сервісів для відеозв'язку та матеріалів

В результаті дослідження платформ-аналогів було сформовано таблицю порівняння особливостей всіх ресурсів та реалізованого програмного продукту.

Таблиця 1.4

Порівняння платформ-аналогів

Функціонал	BUKI	Mathema	ТеМа	Власна розробка
Реєстрація та авторизація користувачів	+	+	+	+
Ролі користувачів	+	+	+	+
Каталог викладачів	+	-	+	+
Самостійний вибір викладача	+	-	+	+

продовж. табл. 1.4

Профілі викладачів	+	+	+	+
Рейтинги та відгуки	+	-	-	+
Пошук викладачів	+	-	+	+
Фільтрація за параметрами	+	-	+	+
Індивідуальні заняття 1:1	+	+	+	+
Вбудоване середовище для занять	-	+	-	-
Система розкладів	-	+	-	-
Облік проведених занять	-	+	-	+
Навчальні матеріали в системі	-	+	-	-
Аналітика прогресу учнів	-	+	-	-
Централізований контроль якості	-	+	-	+
Маркетплейс-модель	+	-	+	+
LMS-функціональність	-	+	-	-
Масштабування кількості викладачів	+	-	+	+

У порівнянні трьох конкурентів більше функціональну компоненту має сервіс Mathema, але під тему проєкту краще підходить BUKI, оскільки він більше орієнтований на вибір різних викладачів, не закритий в одному предметі, та має більшу орієнтацію саме на масштабування кількості акаунтів як учнів, так і вчителів. Попри те Mathema демонструє більше напруження саме у влаштуванні навчального процесу прямо на платформі. ТеМа має схожу структуру як і BUKI, але кількість користувачів набагато менше, що і не дає змогу перевірити її у розрізі зовсім іншого масштабу.

1.2. Постановка задачі

Важливим аспектом розробки є врахування сучасних тенденцій у сфері цифрової освіти. Класичні методи пошуку викладачів часто супроводжуються низкою організаційних проблем: складністю узгодження розкладів між кількома учасниками, відсутністю консолідованої інформації про кваліфікацію спеціаліста та незручністю ведення обліку занять. Вебплатформа покликана нівелювати ці недоліки, створюючи єдине віртуальне середовище, де рутинні організаційні процеси автоматизуються. Це створює умови, за яких викладачі можуть сфокусуватися безпосередньо на навчальному процесі та підготовці матеріалів, а учні отримують гнучкий інструмент для планування свого освітнього шляху відповідно.

Розробка сучасної вебплатформи онлайн-школи, орієнтованої на репетиторство, це комплексне, багатоетапне та високотехнологічне завдання, головною метою якого є створення інноваційного, інтуїтивно зрозумілого та надійного інструментарію для організації освітнього процесу. Платформа має стати централізованим хабом, який об'єднує користувачів з різними потребами, забезпечуючи безперебійний процес пошуку викладачів, запису на заняття, комунікації та загальної взаємодії між учасниками освітнього процесу. Фундаментальною основою проєкту є розробка інтелектуальної системи категоризації та каталогізації вчителів відповідно до їхніх спеціальностей, досвіду та напрямків викладання. Це вимагає впровадження продуманої навігаційної системи, гнучких фільтрів та алгоритмів пошуку, що дозволить користувачам ефективно знаходити потрібних спеціалістів. Не менш важливою складовою є розробка підсистеми модерації та аудиту взаємодій на платформі, управління якою буде здійснюватися адміністрацією школи задля забезпечення високої якості надання освітніх послуг.

Окрему увагу в процесі проєктування необхідно приділити життєвому циклу організації заняття та системі комунікації. Процес взаємодії на платформі не обмежується лише фактом пошуку викладача; він включає складний ланцюжок подій: формування запиту на заняття, етап підтвердження або

відхилення заявки викладачем, можливості перенесення чи скасування зустрічі, а також логіку нагадувань. Реалізація комплексної системи транзакційних сповіщень є критично важливою для підтримки високого рівня залученості користувачів та мінімізації кількості пропущених занять. Цей функціонал вимагає розробки чіткої бізнес-логіки для відслідковування статусів бронювань та їх синхронізації в режимі реального часу.

Базовим архітектурним компонентом вебплатформи є надійна та захищена система авторизації та автентифікації. Ця система повинна підтримувати рольову модель доступу, яка чітко розмежовує права та можливості різних типів користувачів. На початковому етапі кожен відвідувач створює базовий користувацький акаунт, який надалі дозволяє обрати або змінити вектор взаємодії з платформою. Акаунт учня має надавати широкий спектр можливостей: від перегляду розкладу та профілів викладачів до безпосереднього бронювання занять та перегляду історії навчання. З іншого боку, кожен зареєстрований користувач повинен мати технічну можливість ініціювати процес переходу в статус викладача, подавши відповідну стандартизовану заявку. Цей процес передбачає заповнення кваліфікаційної анкети, після чого заявка потрапляє до електронної черги на розгляд адміністрації платформи, яка приймає рішення про затвердження або відхилення кандидатури.

Для забезпечення максимально релевантного вибору викладача, на сайті передбачена багаторівнева система навігації. Користувач зможе здійснювати пошук від загального до конкретного: спочатку обираючи широку категорію дисциплін, потім - конкретний предмет, і, зрештою, застосовуючи додаткові параметри фільтрації. Для підвищення рівня довіри між учасниками платформи та формування об'єктивної оцінки компетенцій спеціалістів, розробляється прозора система відгуків та рейтингів. Вона дозволить учням залишати розгорнуті коментарі та оцінки про викладачів, що стане ключовим фактором при прийнятті рішень новими користувачами. Адміністративний модуль, у свою чергу, надасть вичерпний інструментарій для управління платформою: моніторинг активності користувачів, обробка заявок на зміну ролі та аудит

фінансових транзакцій.

Зважаючи на те, що вебплатформа оброблятиме та зберігатиме значні обсяги конфіденційної інформації, критичним завданням є імплементація сучасних протоколів безпеки. Необхідно реалізувати систему безпечного передавання даних, криптографічного хешування паролів та захисту від поширених вебвразливостей. При цьому, архітектура безпеки не повинна негативно впливати на загальну швидкодію сайту, зберігаючи високі показники продуктивності та забезпечуючи комфортний UI/UX досвід для кінцевого споживача.

Функціональне ядро системи базуватиметься на безперебійному виконанні базових операцій щодо всіх ключових сутностей бази даних. Це стосується управління акаунтами, редагування профілів, створення та модерації відгуків, додавання нових навчальних предметів та категорій. Адміністративна панель повинна стати ізольованим, максимально захищеним середовищем, доступ до якого мають лише авторизовані користувачі з відповідним рівнем привілеїв. Цей модуль забезпечить регуляцію всієї діяльності на сайті та підтримку його життєздатності.

На завершальному етапі необхідно визначити стратегію розгортання проєкту та забезпечення стабільності його роботи в умовах реального навантаження. Перед запуском платформи в експлуатацію має бути проведено комплексне тестування. Воно повинно включати модульне тестування окремих компонентів коду, інтеграційне тестування взаємодії різних модулів системи, наскрізне тестування інтерфейсу, а також навантажувальне та стрес-тестування для перевірки поведінки системи при великій кількості одночасних запитів.

Результатом виконання даної проєктної реалізації стане повноцінна, готова до використання вебплатформа онлайн-школи, побудована з використанням передових технологій. Вона повністю задовольнить поставлені функціональні вимоги, вирішить проблему пошуку та взаємодії між учнями та викладачами, а завдяки продуманій архітектурі, ергономічному UI/UX дизайну та надійному технологічному стеку забезпечить високу ефективність, стабільність та безпеку

в процесі експлуатації.

1.3. Вибір архітектури веборієнтованої системи

Вибір архітектури є фундаментальним етапом розробки проєкту, оскільки він вирішує порядок взаємодії різних компонентів між собою, спосіб передачі даних та відображення інтерфейсу для користувача. Архітектура вирішує на які компонентні структури буде поділено програмну реалізацію та як ці компоненти будуть взаємодіяти між собою. Вдало продумана архітектура дає можливість адекватно масштабувати проєкт згодом, підтримувати його та мінімізує ризики виникнення помилок.

В поточному проєкті було прийнято рішення використовувати структуру Rest API, що дозволяє відокремити структури клієнтської та серверної компоненти, що в довгостроковій перспективі дозволяє змінювати одну з них, не вносячи змін до іншої. В подальшому при потребі можна повністю замінити складову користувацького інтерфейсу у разі появи нових фреймворків, використання яких буде більш доцільним. На основі планування взаємодії компонентів системи було сформовано структурну схему програмного комплексу (рис. 1.4).

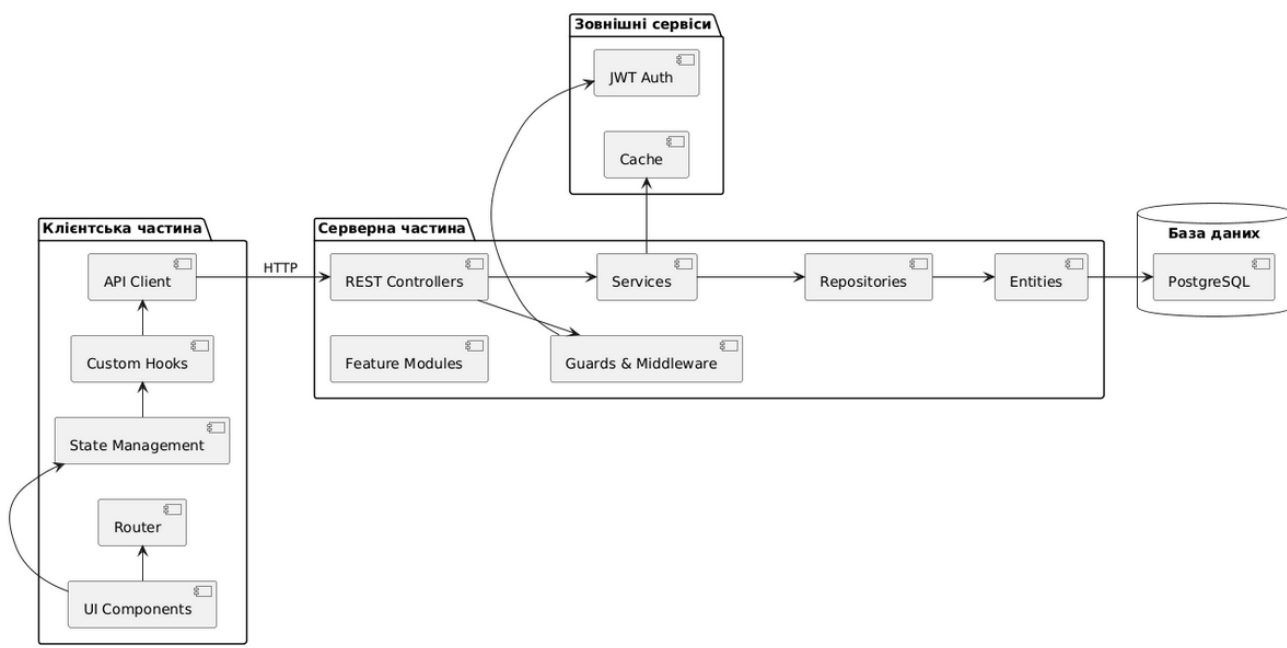


Рис. 1.4 - Структурна схема програмного комплексу

Rest Api впроваджує гнучкість взаємовідношення клієнтської та серверної частини, оскільки серверна складова повертає дані, а не HTML-код, що дозволяє формувати фронтенд структуру незалежно від серверної. Така структура полегшує процес розробки та масштабування проєкту.

NestJS використовує модульну структуру проєкту, що ділить програмний код на модулі відповідно до сутності, з якою відбувається робота. Така структура дозволяє чітко розділити програмний код, що полегшує його перегляд, підтримку та розширення. Даний фреймворк використовує широкий набір інструментів задля роботи із запитами: guards, interceptors, pipes і подібні. Такі технології дозволяють зручно працювати з механізмами JWT-авторизації та автентифікації. Отже поточний фреймворк буде задовольняти відповідні поставлені задачі.

Клієнтська архітектура буде реалізована на основі фреймворку React. Взаємодія між клієнтом та сервером буде відбуватись за допомогою useQuery та модуля Axios, що дозволить органічно реалізовувати асинхронні запити до бекенду. Такий вибір обумовлено можливістю UI/UX-орієнтованого очікування відповіді та обробки помилок. Запити будуть відбуватись через протокол HTTP та будуть проходити перевірку через політику CORS [8].

Архітектурно клієнтська частина побудована за принципом багатошарової архітектури, де кожен рівень виконує свою ізольовану задачу. Базовий шар представлений папкою з API-сервісами на основі Axios, які відповідають за безпосередню комунікацію з сервером. Шар управління станом реалізовано за допомогою бібліотеки TanStack Query, яка забезпечує отримання даних, їх автоматичне кешування та синхронізацію, а також надає зручний інтерфейс для відстеження статусів завантаження та помилок через хук useQuery [15].

Логіка відображення розподілена між двома рівнями: папкою pages, компоненти якої виступають у ролі контролерів, що акумулюють дані для конкретного маршруту, та папкою components, яка містить атомарні елементи інтерфейсу для візуалізації отриманої інформації. Перемикання між функціональними блоками системи забезпечується роутером, який організовує

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						21

навігацію в межах Single Page Application, забезпечуючи цілісність користувацького досвіду без перезавантаження сторінок. Багатошарову архітектуру клієнтської частини наведено у формі діаграми (рис. 1.5).

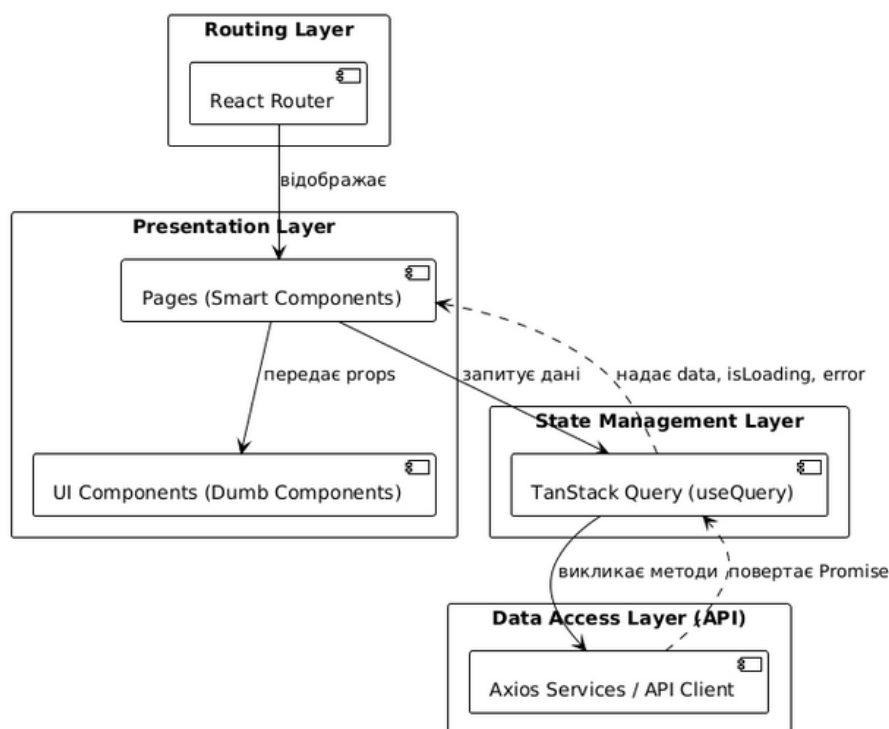


Рис. 1.5 – Візуалізація багатошарової архітектури клієнтської компоненти

Така структура дозволяє логічно поділити код на відповідні секції, що допомагає у підтримці, налагодженні та масштабуванні проєкту. Кожна складова виконує певний тип логіки, що контролює нагромадження коду.

Завдяки чіткому розмежуванню бізнес-логіки і логіки відображення, значно підвищується рівень повторного використання коду. Ізольовані атомарні компоненти інтерфейсу, такі як елементи форм, картки профілів чи навігаційні панелі, можуть багаторазово застосовуватися на різних сторінках платформи без дублювання базового коду. Це не лише зменшує загальний обсяг клієнтського застосунку, але й суттєво спрощує процес рефакторингу чи оновлення візуального стилю. Крім того, виокремлення мережевих запитів у незалежний шар дозволяє легко адаптувати клієнтську частину до можливих змін у контрактах REST API на стороні бекенду без необхідності втручання безпосередньо у візуальні компоненти.

Ще одним важливим аспектом такого архітектурного підходу є ефективне управління складними станами програми, що є критично необхідним з огляду на наявність різних ролей користувачів та їх специфічних інтерфейсів. Використання сучасних інструментів кешування та синхронізації у поєднанні зі строгою типізацією даних під час обміну між клієнтом та сервером гарантує високу передбачуваність роботи інтерфейсу. Такий розподіл відповідальності мінімізує ризик виникнення помилок під час виконання програми та забезпечує плавну, безперебійну взаємодію користувача з платформою, створюючи надійне підґрунтя для подальшого розширення функціоналу.

Висновки до першого розділу

У першому розділі визначено основні завдання та технічні вимоги до розробки вебплатформи онлайн-школи. Проведено аналіз існуючих сервісів, що надають можливість вибору викладача та проведення індивідуальних занять у форматі один на один, з метою визначення їх функціональних можливостей, переваг і недоліків. На основі отриманих результатів сформовано вимоги до функціональності та структури розроблюваної системи. Архітектура платформи побудована за клієнт-серверною моделлю з урахуванням вимог до масштабованості, підтримуваності та безпеки даних.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для реалізації серверної частини було обрано фреймворк NestJS, котрий побудовано на основі Node.js з використанням Express та на базі мови Typescript. Такий підхід дозволяє реалізувати модульну структуру проєкту, що відділить логіку взаємодії з сутностями БД одна від одної. Даний фреймворк надає великий функціонал обробки запитів до потрапляння в контролер, що суттєво допомагає у процесі обробки даних авторизації/автентифікації. Статична типізація допомагає в обробці помилок, що виправляє велику кількість багів, котрі могли б виникнути при використанні Javascript.

Під час розробки клієнтської частини було використано фреймворк React у поєднанні з Typescript. React містить велику кількість модулів, що дає широкий простір вибору як інструментів взаємодії з сервером, так і можливостей верстки сторінок [11]. Такий фреймворк з використанням віртуального DOM значно облегшує промальовку елементів, дані котрих змінюються, бо при зміні одного елемента не перемальовується сторінка в цілому, що зменшує навантаження на браузер [13]. Статична типізація допомагає аналогічним чином як і в серверній частині. Задля відображення стилів був обраний CSS, котрий використовується з приставкою `module` у кожному файлі, задля уникнення конфліктів стилів. Оскільки було обрано шлях стилізації сайту в напрямку жанру коміксів, то й було вирішено писати стилі з нуля [9].

Взаємодія між клієнтом та сервером відбувається за допомогою хука `useQuery` та модуля `Axios`. `Axios` реалізує сам запит та налаштування його, а саме базове посилання до серверу, додаткові дані вшиті в нього (токени та заголовки) та тип методу запиту [4]. `useQuery` ж відповідає за асинхронну складову запитів, а саме логіку відображення сторінки під час очікування на відповідь від сервера, або відображення повідомлення у разі помилок, що забезпечує сайт безперебійністю роботи у разі повернення некоректних даних. Для відображення повідомлення про помилки взаємодії клієнта та сервера використано бібліотеку

react-hot-toast, котра дозволяє вивести повідомлення помилки, котре передає сервер, що покращує UI/UX сайту і дає змогу користувачу зрозуміти, що саме пішло не так.

Як БД було обрано PostgreSQL, оскільки вона має великий список ORM, котрі з нею можуть взаємодіяти. Така БД має зручний GUI додаток через який зручно перевіряти поточний стан таблиць та даних. Така БД забезпечує широку інтеграцію з типами даних, котрі використовує Typescript через конвертацію за допомогою ORM. БД забезпечує швидкодію, використання великого об'єму даних та безпеку проведення транзакцій за допомогою вбудованого інструментарію авторизації/автентифікації задля доступу до даних.

Було обрано реляційну структуру даних, бо це забезпечує логіку взаємодії таблиць між собою, а саме влаштування зв'язків. Оскільки буде використано велику кількість різнотипних акаунтів, котрі будуть взаємопов'язаними, то необхідно відбудувати логіку зв'язків між ними, що і забезпечує реляційна структура. Така структура дозволяє згодом виконувати складні запити пов'язані з таблицями, котрі пов'язані між собою, отже такий тип підходить до поточного проєкту.

Як ORM було обрано TypeORM, через хорошу конвертацію типів з Typescript в PostreSql. TypeORM має широку взаємодію з NestJS, а саме велику кількість методів, котрі допомагають взаємодіяти з БД без використання SQL-запитів. Побудова таблиць і їх зв'язків не вимагає додаткового опрацювання окремо в БД. Використання такого інструментарію значно зменшує кількість програмного коду, додає ефективності та структурованості коду.

Для забезпечення надійного збереження та швидкого доступу до мультимедійного контенту (фотографій профілів викладачів та скріншотів проведених занять) у межах платформи було обрано інтеграцію з хмарною інфраструктурою Firebase, зокрема використано рішення Firestore. Однією з ключових переваг такого підходу є надання доступу до графічних матеріалів за прямим посиланням безпосередньо з хмарного ресурсу [5]. Завдяки цьому клієнтський застосунок завантажує зображення оминаючи основний сервер

додатків, що суттєво знижує навантаження на нього та пропускну здатність мережі. Таке архітектурне рішення мінімізує час очікування для кінцевого користувача, забезпечує високу доступність контенту та дозволяє ефективно використовувати механізми кешування браузера.

Важливим критерієм вибору наведеного технологічного стеку є також його висока пристосованість до впровадження сучасних практик автоматизованого контролю якості програмного забезпечення. Завдяки наскрізній строгій типізації за допомогою TypeScript як на клієнті, так і на сервері, створюється надійне підґрунтя для проведення глибокого інтеграційного та наскрізного тестування користувацького інтерфейсу. Ізольованість компонентів React та чітко специфікована структура API дозволяють ефективно симулювати дії користувачів та перевіряти коректність роботи UI-сценаріїв. Водночас модульна побудова серверної частини полегшує підготовку платформи до етапів навантажувального та стрес-тестування, що є критично необхідним кроком для перевірки стабільності системи та оцінки її здатності витримувати велику кількість одночасних транзакцій від учнів та вчителів.

Окрім того, використання екосистеми React та NestJS закладає міцний фундамент для потенційного розширення продукту в майбутньому. У разі виникнення потреби у розробці мобільних застосунків для онлайн-школи, наявна архітектура та обрані патерни дозволять максимально ефективно перевикористати існуючу логіку взаємодії з сервером. Уніфікація розробки навколо TypeScript суттєво знижує технічні бар'єри для реалізації кроссплатформних мобільних рішень, оскільки значна частина структур даних, інтерфейсів передачі даних та загальної бізнес-логіки вже стандартизована та відлагоджена в межах вебверсії. Це стратегічне рішення оптимізує ресурси на подальшу підтримку та еволюцію проєкту.

В результаті вибору технологічного стеку було обрано низку сучасних програмних інструментів, які забезпечують можливість масштабування системи, її довготривалу підтримку та належний рівень безпеки даних. Обрані технології відповідають актуальним вимогам до розробки вебзастосунків і широко

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						26

використовуються у комерційних програмних продуктах, що підтверджує їх надійність та ефективність. Такий підхід дозволяє гнучко адаптувати систему до зростання кількості користувачів і функціональних можливостей, а також спрощує процес супроводу та оновлення програмного забезпечення. Крім того, використаний технологічний стек створює передумови для подальшого розширення проєкту, зокрема шляхом інтеграції нових фреймворків, бібліотек і модулів без необхідності суттєвої зміни існуючої архітектури системи.

2.2. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Процес аналізу вимог передбачає ідентифікацію та детальну специфікацію функціональних і нефункціональних характеристик програмного забезпечення. Даний етап включає дослідження часових показників відгуку системи на запити користувачів, моделювання сценаріїв експлуатації, а також розробку архітектурної моделі компонентів із визначенням їхньої структури та принципів взаємодії. Такий підхід забезпечує комплексне розуміння потреб цільової аудиторії та встановлення технічних параметрів, необхідних для створення конкурентоспроможного програмного продукту.

Функціональні вимоги визначають сукупність сервісів та операцій, які система повинна виконувати для реалізації свого цільового призначення та повного задоволення запитів кінцевих користувачів.

Відповідно до архітектури вебплатформи, передбачено розмежування прав доступу за допомогою таких рольових моделей:

1. Користувач це зареєстрований суб'єкт, який пройшов первинну ідентифікацію та має базовий набір прав для взаємодії з інтерфейсом.
2. Учень це роль, що надає доступ до освітнього контенту, функцій запису на заняття та проведення фінансових операцій.
3. Викладач це роль, призначена для організації навчального процесу, ведення звітності та отримання виплат за надані послуги.

4. Модератор це користувач, відповідальний за дотримання правил платформи та моніторинг активності інших учасників.
5. Адміністратор це роль з високим рівнем доступу, що забезпечує керування кадровим складом і наповненням сутностей предметів і категорій.
6. Власник це суб'єкт з максимальним рівнем повноважень, що здійснює стратегічне управління платформою та фінансовий аудит.

Для зареєстрованого користувача система надає можливість заповнення персональних даних, вибору вектору подальшої взаємодії з ресурсом та створення власного учнівського профілю або подання заявки на працевлаштування. Окрім цього, реалізовано механізм подання запитів на отримання розширених прав доступу.

Після активації ролі Учня, користувач отримує доступ до функціоналу вибору викладачів та запису на навчальні заняття. Система забезпечує можливість здійснення безготівкових платежів, редагування спеціалізованого профілю учня, а також використання модуля зворотного зв'язку для залишення відгуків про роботу педагогічного складу.

Викладач наділений інструментарієм для створення та управління сутностями уроків. Функціонал цієї ролі передбачає ведення електронної звітності за результатами проведених занять, можливість оцінювання діяльності учнів через систему відгуків та управління власним професійним профілем.

Діяльність Модератора спрямована на забезпечення якості контенту та дотримання етики спілкування. Він має доступ до перегляду відгуків користувачів і звітів про уроки, наділений правом застосування дисциплінарних заходів та може ініціювати запит на підвищення рівня прав до статусу адміністратора.

Адміністратор здійснює управління операційними процесами платформи. До його компетенцій належить розгляд заявок на отримання ролей викладача та модератора, керування переліком навчальних дисциплін, а також контроль за діяльністю модераторів із можливістю відкликання їхніх повноважень.

Найвищий рівень доступу закріплено за Власником. Ця роль дозволяє формувати структуру платформи шляхом створення нових категорій, приймати рішення щодо призначення адміністраторів та здійснювати комплексний аудит фінансових потоків і оплат, проведених у межах системи.

Нефункціональні вимоги регламентують властивості та якісні показники системи, що визначають її ефективність, надійність та експлуатаційну придатність. Для забезпечення високої якості освітньої платформи було визначено наступні ключові аспекти:

Інтерфейс платформи має відповідати принципам інтуїтивності, мінімізуючи час на адаптацію користувача до системи. Пріоритетом є реалізація дизайну, що гарантує коректне відображення та функціонування.

Система повинна демонструвати високу швидкість обробки запитів: час відгуку для стандартних операцій не має перевищувати 1.5–2 секунд, а для складних транзакцій - до 5 секунд. Архітектура ПЗ має забезпечувати стабільну роботу за умови одночасної активності до 300 користувачів, зберігаючи цілісність даних та швидкість виконання запитів під час пікових навантажень.

Захист персональної інформації та фінансових транзакцій є критичним аспектом. Усі дані мають передаватися виключно через захищений протокол HTTPS з використанням шифрування TLS. Зберігання паролів користувачів у базі даних повинно здійснюватися у вигляді хеш-значень із застосуванням сучасних криптографічних алгоритмів та використанням механізму хешування. Авторизація сесій має базуватися на захищених токенах, що унеможливорює несанкціонований доступ до облікових записів.

Вебплатформа повинна демонструвати стабільну та ідентичну роботу в актуальних версіях провідних браузерів: Google Chrome, Apple Safari, Mozilla Firefox та Microsoft Edge. Окрім того, програмний продукт має бути сумісним із різними операційними системами, що забезпечує безбар'єрний доступ до освітніх послуг для всіх категорій користувачів.

Для забезпечення комфортної роботи користувачів платформа має відповідати чітким критеріям швидкодії, де відображення основних елементів

інтерфейсу, навігація між розділами та видача результатів пошуку з фільтрацією виконуються в межах 1–1.5 секунди. Операції, що потребують обробки та збереження даних - зокрема авторизація, зміна ролей, створення нових уроків чи оновлення профілів - мають завершуватися за час до 2 секунд. Більш складні системні процеси вимагають трохи більше часу: обробка фінансових транзакцій із підтвердженням оплати обмежена 3–5 секундами, а формування аналітичних звітів за обраний період - 7 секундами. Щодо роботи з медіафайлами, система повинна відгукуватися на старт завантаження матеріалів не довше ніж за 1 секунду, після чого швидкість передачі даних залежатиме виключно від пропускної здатності інтернет-каналу самого користувача. Дотримання цих жорстких часових лімітів є критично важливим фактором для утримання аудиторії, оскільки мінімізація технічних затримок безпосередньо впливає на задоволеність робочим процесом як учнів, так і викладачів. Крім того, гарантування такої стабільної швидкодії вимагає впровадження ефективних механізмів оптимізації, що дозволить платформі зберігати високу продуктивність навіть у періоди пікових навантажень чи різкого збільшення кількості одночасних сесій.

Специфікація варіантів використання є фундаментальним етапом проєктування освітньої вебплатформи, оскільки вона дозволяє формалізувати сценарії взаємодії між різними категоріями користувачів та програмним комплексом. Даний підхід забезпечує чітке розуміння того, як система підтримує навчальні та адміністративні процеси для досягнення цілей кожного учасника - від учня до власника ресурсу.

Для візуалізації функціональної структури та взаємозв'язків між акторами та прецедентами розроблено діаграму варіантів використання, яку наведено на рисунку нижче (рис. 2.1). Вона дозволяє наочно продемонструвати ієрархію прав доступу та логіку сервісів платформи.

Нижче наведено деталізований опис основних сценаріїв використання системи відповідно до визначених рольових моделей (табл. 2.1).

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						30

Виявлення варіантів використання

Основний актор	Найменування	Формулювання
Користувач	Реєстрація	Цей варіант дозволяє створити власний профіль задля подальшого використання вебплатформи.
	Авторизація	Цей варіант дозволяє увійти у власний профіль задля подальшого використання вебплатформи.
	Перегляд предметів/вчителів	Цей варіант дозволяє користувачеві переглянути які предмети викладаються та які вчителі їх викладають.
	Створення акаунта учня	Цей варіант дозволяє створити користувачеві профіль учня задля подальшого навчання.
	Подання заявки на роль вчителя	Цей варіант дозволяє користувачеві подати заявку, щоб стати вчителем.
	Подання заявки на роль модератора	Цей варіант дозволяє користувачеві подати заявку, щоб стати модератором.
Учень	Запис на урок	Цей варіант дозволяє користувачеві обрати відповідного вчителя і предмет і записатись до нього на заняття.
	Оплата	Цей варіант дозволяє користувачеві оплатити урок прямо через вебплатформу.
	Зміна профілю	Цей варіант дозволяє користувачу змінювати власний учнівський профіль.

Вчитель	Додавання вільних годин	Цей варіант дозволяє вчителю додати у власний графік годин, коли він може працювати.
	Зміна профілю	Цей варіант дозволяє користувачу змінювати власний вчительський профіль.
	Ведення звітності занять	Формування звітності за кожен проведений урок
	Формування сутності уроку	Задання додаткової інформації корисної учню до сутності уроку
	Перегляд відгуків користувачів	Цей варіант передбачає, що модератор переглядає відгуки користувачів і оцінює їх.
Модератор	Перегляд скарг	Цей варіант передбачає, що модератор переглядає скарги надіслані різними групами користувачів.
	Бан користувачів	Цей варіант передбачає, що модератор блокує можливість використання вебплатформи користувачам, котрі порушили її правила.
Адміністратор	Подача заявки на роль адміністратора	Цей варіант передбачає, що модератор може надіслати заявку на отримання ролі адміністратора.
	Розгляд заявок на роль модератора	Цей варіант передбачає, що користувач переглядає надіслані заявки на роль модератора і надає роль у позитивному випадку.

Адміністратор	Розгляд заявок на вакансію вчителя	Цей варіант передбачає, що користувач переглядає надіслані заявки на вакансію вчителя і надає роль у позитивному випадку.
Власник	Перегляд дій модераторів	Цей варіант передбачає, що користувач переглядає аудит дій модераторів.
	Зняття ролі модератора	Цей варіант передбачає, що користувач у разі негативної роботи модератора знімає з нього цю роль.
	Розгляд заявок на роль адміністратора	Цей варіант передбачає, що користувач переглядає надіслані заявки на роль адміністратора і надає роль у позитивному випадку.
	Аудит оплат	Цей варіант передбачає, що користувач переглядає журнал оплат.
	Перегляд дій адміністраторів	Цей варіант передбачає, що користувач переглядає аудит дій адміністраторів.
	Зняття ролі адміністратора	Цей варіант передбачає, що користувач у разі негативної роботи адміністратора знімає з нього цю роль.

На основі наведеної таблиці було сформовано відповідно діаграму Use-case (рис. 2.1).

Отже, побудова діаграми варіантів використання виступає критично важливим етапом у процесі проєктування програмного забезпечення. Даний інструмент дозволяє систематизувати вимоги кінцевих користувачів та трансформувати їх у конкретні функціональні завдання. Аналіз сценаріїв взаємодії на ранніх стадіях розробки сприяє мінімізації проєктних ризиків,

запобігає виникненню логічних помилок та гарантує повну відповідність розробленого функціоналу встановленим технічним специфікаціям.

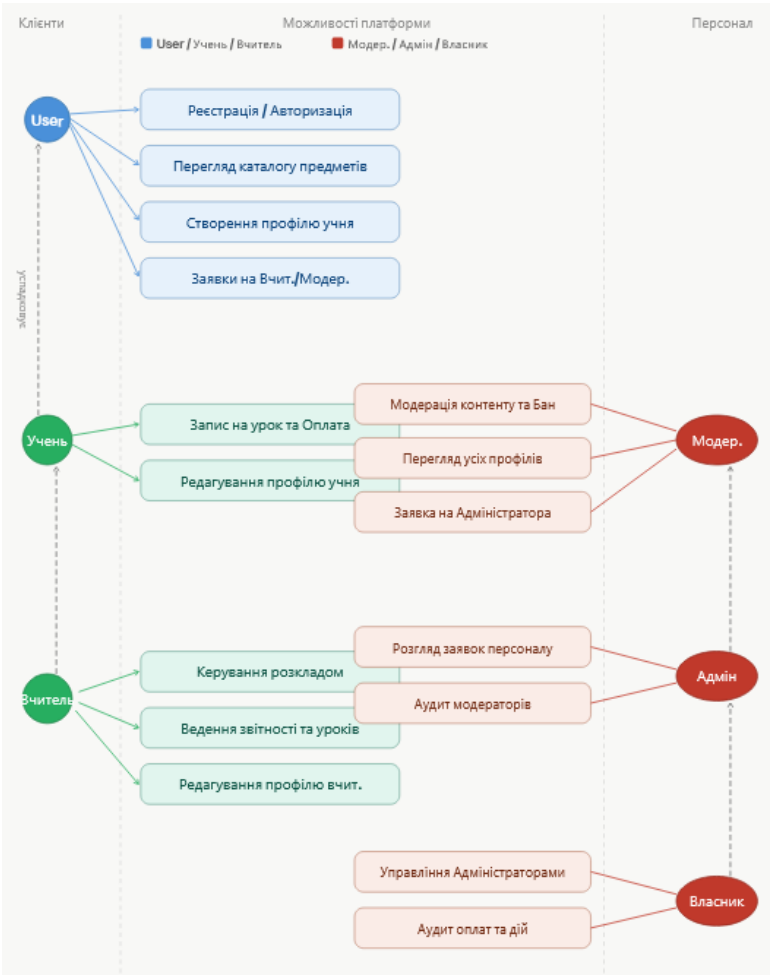


Рис. 2.1 - Діаграма Use-case

Через специфіку технології NestJS структура коду буде поділена на модулі, котрі мають внутрішню взаємодію і відповідають за логіку роботи з конкретною сутністю бази даних [10]. Кожен модуль містить файли таких типів: модуль, сервіс, контроллер, DTO, сутність. Модуль це файл логіки взаємодії з іншими залежностями, реалізує взаємодію компонентів. Сервіс це файл, котрий реалізує взаємодію з базою даних. Контролер це файл для обробки ендпоінтів. DTO це файли, котрі прописують типізацію об’єктів, котрі будуть відправлені/отримані відповідним методом. Сутність це файл котрий відображує структуру відповідної таблиці в базі даних. Така структура дозволяє гнучко та зручно розширювати і масштабувати наявний функціонал системи.

```
classDiagram
    class InitialStudentDto {
        fullName: string
        dateOfBirth: string
        learningGoal: string
    }
    class StudentsController {
        studentsService: StudentsService
        findAll()
        create()
        update()
        delete()
        findStudentByUserId()
        findStudentProfileByUserId()
        getMyTeachers()
    }
    class StudentsGuard {
        studentsService: StudentsService
        canActivate() bool
    }
    class StudentsService {
        studentsRepo: Repository
        userService: UserService
        findAll()
        findStudentByUserId()
        createStudentProfile()
        updateStudentProfile()
        deleteStudentProfile()
        findStudentProfileByUserId()
        findMyTeachersViaService()
        updateRating()
    }
    class UserRepository {
        <<UserModule>>
        <<UserService>>
    }
    class DatabaseRepository {
        <<Database>>
        <<Repository>>
    }
    class EntityStudent {
        id: number
        userId: number
        fullName: string
        dateOfBirth: string
        learningGoal: string
        rating: number
    }

    InitialStudentDto --> StudentsController : Body
    StudentsController --> StudentsGuard : UseGuard
    StudentsGuard --> StudentsService : Inject
    StudentsService --> UserRepository : Inject
    StudentsService --> DatabaseRepository : Inject
    DatabaseRepository --> EntityStudent : CRUD
```

Діаграма відображає архітектуру NestJS-модуля для управління студентами. Запит надходить через `StudentsController`, який приймає тіло запиту у вигляді `InitialStudentDto` і делегує виконання до `StudentsService`. Перед обробкою запит проходить через `StudentsGuard` - захисний механізм, що перевіряє права доступу через метод `canActivate()` і також використовує `StudentsService`. Сам `StudentsService` містить основну бізнес-логіку і взаємодіє з двома залежностями: `UserService` для роботи з користувачами та `Repository` для виконання CRUD-операцій над сутністю `Student`, яка зберігає поля `id`, `userId`, `fullName` та `balance`.

Модуль user - забезпечує управління базовими обліковими записами користувачів. Він регламентує структуру даних акаунта, реалізує CRUD-операції сутності користувача та зберігає загальну інформацію, необхідну для ідентифікації суб'єкта в системі.

Модуль auth - реалізує комплексний механізм автентифікації та авторизації. Модуль описує логіку реєстрації нових учасників, процедуру входу

до системи, підтримку багатофакторної автентифікації через зовнішні сервіси та контроль доступу на основі ролей.

Модуль token - виокремлює логіку оперування токенами доступу з метою підвищення безпеки та читабельності коду. Модуль відповідає за генерування, хешування, валідацію та збереження токенів, забезпечуючи безпечне підтримання користувацьких сесій.

Модуль teacher-profile - призначений для управління професійним профілем викладача. Він дозволяє користувачу з відповідною роллю редагувати інформацію про кваліфікацію, регалії, досвід роботи та досвід викладання, а також керувати публічною видимістю власного профілю.

Модуль student - забезпечує функціонування профілю учня. Модуль реалізує логіку взаємодії користувача з його учнівськими даними, дозволяє відстежувати історію навчання та виконувати операції з налаштування персональних параметрів взаємодії з платформою.

Модуль subject - відповідає за управління переліком навчальних дисциплін. Функціонал модуля дозволяє адміністраторам та власникам платформи створювати, модифікувати та класифікувати предмети, які згодом використовуються вчителями для створення занять.

Модуль lesson - є центральним компонентом системи, що описує структуру та життєвий цикл навчального заняття. Модуль регламентує взаємодію викладача з уроком, інтегрується з модулем звітності та ініціює запити до фінансового модуля для проведення розрахунків.

Модуль payment - реалізує логіку фінансових транзакцій у межах платформи. Він забезпечує проведення оплат за уроки, фіксацію статусів платежів та зберігання історії транзакцій для проведення аудиту власником.

Модуль review - забезпечує функціонування системи зворотного зв'язку. Модуль дозволяє учням залишати оцінки та текстові відгуки про вчителів, а вчителям - оцінювати прогрес учнів. Дані модуля використовуються для розрахунку динамічного рейтингу викладачів.

Модуль application - відповідає за процес масштабування прав користувачів. Модуль обробляє заявки на отримання ролей вчителя, модератора або адміністратора, забезпечуючи передачу запитів на розгляд відповідним керівним ролям.

Модуль firestore - реалізує логіку роботи з файловими об'єктами. Модуль забезпечує взаємодію та підключення до сервісів системи Firebase, взаємодію інших таблиць із записами зображень.

Модуль mail - реалізує логіку роботи з відправленням листів користувачам, в ньому відбувається підключення сервісів розсилки та реалізацію шаблонів листів.

Таким чином, розроблена модульна архітектура серверної частини формує надійний фундамент для стабільного функціонування освітньої платформи. Чіткий розподіл відповідальності між модулями забезпечує високу відмовостійкість системи: потенційні зміни або помилки в одному сегменті не дестабілізують роботу всього застосунку. Запропонована структура дозволяє не лише ефективно реалізувати поточні бізнес-вимоги - такі як керування навчальними планами, проведення безпечних транзакцій та модерація контенту, а й закладає потенціал для горизонтального масштабування та легкої інтеграції нових сервісів у майбутньому.

2.3. Розробка бази даних системи

Для зберігання та структурування інформації в межах освітньої платформи обрано PostgreSQL - одну з реляційних систем керування базами даних з відкритим вихідним кодом. Даний вибір зумовлений потребою в забезпеченні високої надійності, підтримці ACID-транзакцій та здатності ефективно обробляти складні зв'язки між об'єктами системи, що є критичним для платформ із фінансовими операціями та навчальними графіками.

PostgreSQL вирізняється стабільністю при високих навантаженнях та широкими можливостями масштабування [12]. Важливою перевагою для розробки на сучасному технологічному стеку є безшовна інтеграція СУБД із

ORM TypeORM, що дозволяє описувати схеми даних безпосередньо у коді застосунку, забезпечуючи типізацію та спрощуючи маніпуляцію даними.

У межах розробленого проєкту освітньої платформи PostgreSQL використовується для підтримки розгалуженої моделі даних, що охоплює:

- облікові записи користувачів та механізми автентифікації;
- спеціалізовані профілі викладачів та учнів;
- реєстр навчальних дисциплін та категорій;
- розклад та параметри проведення занять;
- фінансову історію оплат та аудиторські логи.

Використання жорстких обмежень цілісності гарантує високу консистентність даних, запобігаючи виникненню логічних невідповідностей. Для оптимізації швидкодії складних запитів, пов'язаних із фільтрацією вчителів за багатьма критеріями або генерацією звітів, застосовано механізми індексування.

Для повноцінного функціонування сервісу було спроектовано 12 реляційних таблиць, які забезпечують структуроване збереження інформації: user, token, teacher_profile, student, subject, lesson, payment, review, role_application, lesson_report, audit_log.

Така структура бази даних дозволяє ефективно оперувати пов'язаними сутностями, забезпечує масштабованість архітектури та є надійним підґрунтям для реалізації повного спектра функціональних можливостей освітнього сервісу.

Таблиця 2.2

Опис таблиці user

Назва	Тип	Опис
id	integer	Ідентифікатор користувача
username	text	Користувацьке ім'я, котре користувач сам собі обирає
password	text	Захешований пароль користувача
email	text	Електронна скринька користувача

продовж. табл. 2.2

status	text	Відображує поточний статус користувача, активний/заблокований/неактивний
lastLogin	timestamp without timezone	Поле для відображення останнього входу в систему користувачем
roles	text[]	Список ролей поточного користувача
phoneNumber	text	Номер телефону користувача
token	Token	Токен для авторизації із зв'язком один до багатьох
googleId	text	Ідентифікатор користувача в системі Google для OAuth(опціональне поле, котре використовується у випадку використання Google- OAuth)
createdAt	timestamp	Дата та час створення облікового запису
isEmailVerified	boolean	Статус підтвердження електронної скриньки

User – таблиця котра несе в собі основну базову інформацію про користувача ролі Користувач, містить зв'язок з таблицею token один до багатьох.

Таблиця 2.3

Опис таблиці token

Назва	Тип	Опис
id	integer	Унікальний ідентифікатор токена
user	User	Зв'язок багато до одного з таблицею користувача
refreshTokenHash	text	Захешований refresh token
accessTokenHash	text	Захешований access token
createdAt	timestamp	Час створення токена
expiresAt	timestamp	Час зупинки дії токена

Token - таблиця, що зберігає токен для авторизації користувачів, має зв'язок багато до одного з таблицею user.

Таблиця 2.4

Опис таблиці teacher_profile

Назва	Тип	Опис
id	integer	Унікальний ідентифікатор вчителя
fullName	text	Повне ім'я вчителя
bio	text	Коротка біографія вчителя
degree	text	Освіта вчителя
experience	text	Досвід роботи вчителя
pricePerHour	numeric(10, 2)	Ціна роботи вчителя за годину роботи часу
subjects	Subject[]	Зв'язок багато до багатьох з таблицею предметів
students	Student[]	Зв'язок багато до багатьох з таблицею учнів
availableTimes	text[]	Список вільних годин, в котрі вчитель може працювати
userId	integer	Ідентифікатор користувача – власника поточного акаунта
rating	float	Поле, котре показує середній рейтинг по відгукам
languages	text[]	Список мов, на яких викладає вчитель
levels	text[]	Рівні, на яких викладає вчитель предмети
imageUrl	text	Посилання на фото профілю вчителя

Teacher_profile - таблиця, що відображує профіль користувача ролі Вчитель, відображує його конкретні професійні релагії, можливості викладання. Реалізовано декілька масивав з відображенням списків системи.

Таблиця 2.5

Опис таблиці student

Назва	Тип	Опис
id	integer	Унікальний ідентифікатор учня
fullName	text	Повне ім'я учня
dateOfBirth	timestamp	Дата народження учня для розуміння його віку
learningGoal	text	Ціль навчання учня, котру він сам вказує
pricePerHour	numeric(10, 2)	Ціна роботи вчителя за годину роботи часу
teachers	TeacherProfile[]	Зв'язок багато до багатьох з таблицею вчителів
userId	integer	Ідентифікатор користувача – власника поточного акаунта

Student - таблиця, що відображує профіль користувача ролі учня, його потреби, короткий опис, інформацію про баланс на гаманці платформи.

Таблиця 2.6

Опис таблиці subject

Назва	Тип	Опис
id	integer	Унікальний ідентифікатор навчального предмета
name	text	Назва навчального предмета
description	text	Короткий опис поточної дисципліни
icon	text	Захешована іконка предмету, переформатована в рядок
category	text	Категорія області наук предмету
level	text	Рівень викладання предмету

продовж. табл. 2.6

teachers	TeacherProfile[]	Зв'язок багато до багатьох з таблицею вчителів
----------	------------------	--

Subject - таблиця, що відображує навчальний предмет, а саме його опис, назву, короткий опис та відображення.

Таблиця 2.7

Опис таблиці lesson

Назва	Тип	Опис
id	integer	Первинний ключ, унікальний ідентифікатор заняття.
teacherId	integer	Зовнішній ключ, що посилається на teacher_profile
studentId	integer	Зовнішній ключ, що посилається на student
subjectId	integer	Зовнішній ключ, що посилається на subject
topic	text	Тема конкретного заняття, яку вказує викладач при плануванні
startTime	timestamp	Дата та точний час початку проведення уроку
endTime	timestamp	Дата та точний час завершення проведення уроку
meetingUrl	text	Посилання на платформу відеозв'язку
priceAtBooking	numeric	Фіксована ціна за годину на момент бронювання
homework	text	Опис або посилання на домашнє завдання для учня

продовж. табл. 2.7

isPaid	boolean	Прапорець, що вказує, чи було заняття успішно оплачене
createdAt	timestamp	Дата створення запису про урок у системі

Сутність lesson регламентує процес взаємодії між викладачем та учнем у межах освітньої платформи. Вона виконує роль сполучної ланки між трьома ключовими акторами: учнем, вчителем та адміністрацією.

Таблиця 2.8

Опис таблиці payment

Назва	Тип	Опис
id	integer	Первинний ключ, унікальний ідентифікатор транзакції
lessonId	integer	Зовнішній ключ, що пов'язує оплату з конкретним заняттям у таблиці lesson
userId	integer	Зовнішній ключ, що вказує на платника
subjectId	integer	Зовнішній ключ, що посилається на subject
amount	numeric	Сума платежу
currency	text	Валюта операції
status	text	Поточний статус транзакції
transactionId	text	Унікальний ідентифікатор операції у зовнішній платіжній системі
paymentMethod	text	Спосіб оплати
createdAt	timestamp	Дата та точний час ініціалізації платежу
updatedAt	timestamp	Дата останньої зміни статусу транзакції

Сутність payment призначена для фіксації та моніторингу всіх фінансових потоків у межах платформи. Вона забезпечує юридичне та технічне підтвердження факту надання послуг і є основою для формування фінансової звітності власника сервісу. Використовується у поєднанні з двома сервісами

оплати, а саме Stripe та WayForPay. Чітко під'єднадна зв'язком один-до-одного з сутністю конкретного уроку.

Таблиця 2.9

Опис таблиці review

Назва	Тип	Опис
id	integer	Первинний ключ, унікальний ідентифікатор відгуку
teacherId	integer	Зовнішній ключ, що вказує на викладача, якому адресовано відгук
studentId	integer	Зовнішній ключ, що вказує на автора відгуку
rating	integer	Числова оцінка якості викладання
comment	text	Текстовий зміст відгуку з враженнями учня про процес навчання
isModerated	boolean	Статус проходження модерації
createdAt	timestamp	Дата та час публікації відгуку

Сутність review виступає основним інструментом формування соціального капіталу та прозорості освітнього процесу на платформі. Вона реалізує механізм зворотного зв'язку, дозволяючи учням оцінювати професійну компетенцію викладачів та якість проведених занять.

Таблиця 2.10

Опис таблиці role_application

Назва	Тип	Опис
id	integer	Первинний ключ, унікальний ідентифікатор заявки
userId	integer	Зовнішній ключ, що вказує на користувача, який подав запит на зміну ролі

продовж. табл. 2.10

requestedRole	text	Назва ролі, на яку претендує користувач
status	text	Поточний стан розгляду запиту
comment	text	Додаткова інформація від користувача
rejectionReason	text	Причина відмови у наданні ролі
reviewerId	integer	Зовнішній ключ, що вказує на адміністратора або власника, який опрацював заявку
createdAt	timestamp	Дата та час подання заявки користувачем
updatedAt	timestamp	Дата та час останньої зміни статусу

Сутність role_application реалізує бізнес-логіку верифікації та розширення прав доступу в межах освітньої системи. Вона виступає посередником у процесі кадрового наповнення платформи, забезпечуючи контрольований перехід користувача до ролей із вищим рівнем відповідальності.

Таблиця 2.11

Опис таблиці lesson_report

Назва	Тип	Опис
id	integer	Первинний ключ, унікальний ідентифікатор звіту
lessonId	integer	Зовнішній ключ, що забезпечує зв'язок один до одного з конкретним заняттям у таблиці lessonp
teacherId	integer	Зовнішній ключ, що вказує на автора звіту
content	text	Детальний опис проведеної роботи, вивчених тем та досягнутих результатів
studentProgress	text	Оцінка прогресу учня за результатами конкретного уроку

продовж. табл. 2.11

attendance	boolean	Показник присутності учня на занятті
homeworkStatus	text	Статус виконання попереднього домашнього завдання
recommendations	text	Поради викладача щодо подальшого навчання або наступних кроків для учня
createdAt	timestamp	Дата та час формування звіту в системі

Сутність lesson_report призначена для документування результатів освітнього процесу та забезпечення прозорості взаємодії між викладачем та учнем. Вона виступає офіційним підтвердженням виконання навчального плану та є основним джерелом даних для аналізу успішності.

Таблиця 2.12

Опис таблиці audit_log

Назва	Тип	Опис
id	integer	Первинний ключ, унікальний ідентифікатор запису в журналі аудиту
userId	integer	Зовнішній ключ, що вказує на ініціатора дії
action	text	Назва виконаної операції
entityName	text	Назва сутності, до якої були внесені зміни
entityId	integer	Унікальний ідентифікатор конкретного об'єкта, над яким проводилася дія
oldValue	jsonb	Стан об'єкта до виконання операції
newValue	jsonb	Стан об'єкта після виконання операції
ipAddress	text	Мережева адреса, з якої було надіслано запит на виконання дії
createdAt	timestamp	Точна дата та час фіксації події в системі

Сутність audit_log реалізує механізм системного журналювання та ретроспективного аналізу дій користувачів. Вона є фундаментальним інструментом забезпечення підзвітності та безпеки освітньої платформи. На

відміну від звичайних логів застосунку, журнал аудиту зберігає бізнес-контекст змін: хто, коли та які саме дані модифікував.

Результатом проєктування БД є відповідна діаграма (рис. 2.3-2.5) з відображенням взаємодії таблиць.

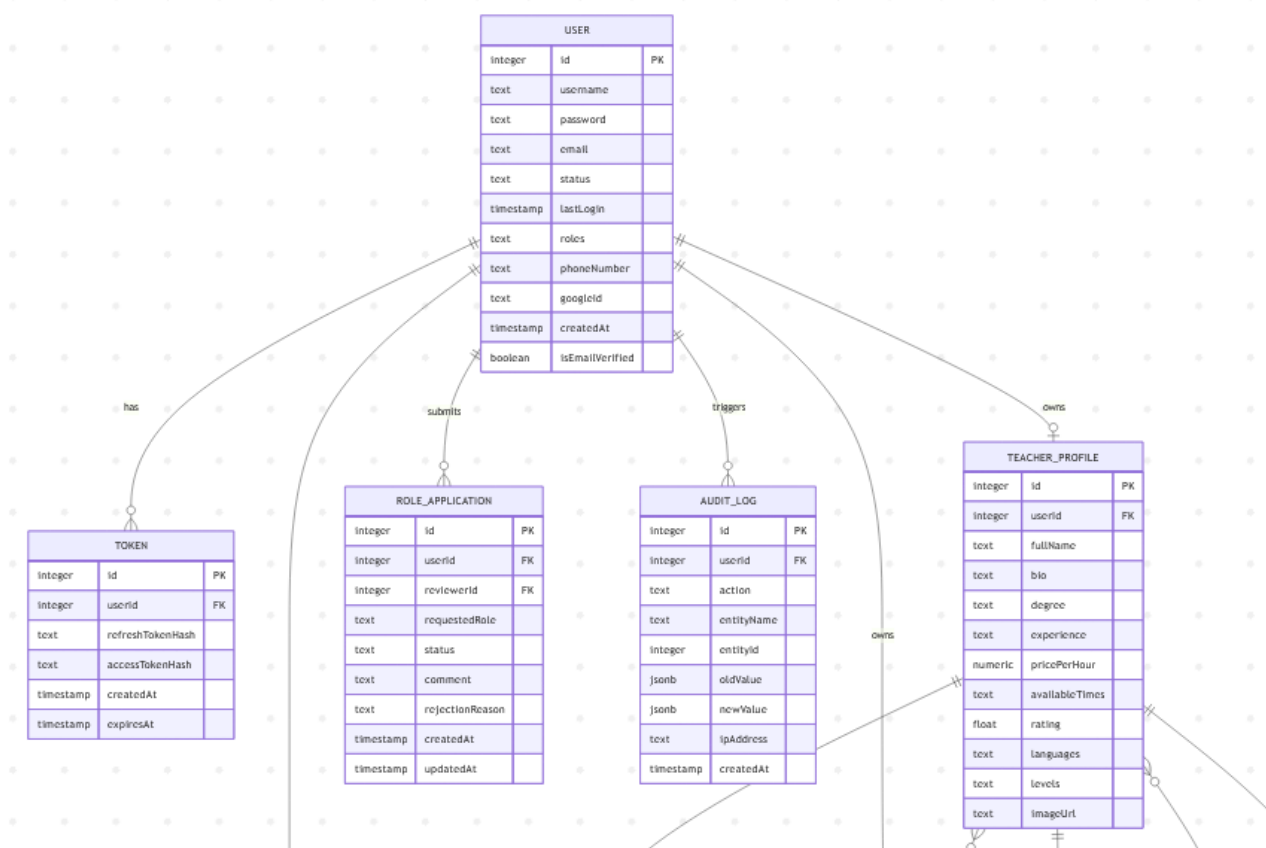


Рис. 2.3 – Перша частина діаграми класів

На цій частині діаграми зображено центральну сутність USER (рис. 2.3), яка є основою всієї системи та містить базові дані користувача: облікові дані, статус, час останнього входу, ролі та прив'язку до Google-акаунту. Від USER розходяться чотири зв'язки: до TOKEN до ROLE_APPLICATION, до AUDIT_LOG (журнал усіх дій користувача в системі з фіксацією старих і нових значень та IP-адреси) і до TEACHER_PROFILE (профіль викладача з детальною інформацією про досвід, ціну за годину, доступний час, мови та рівні навчання). Всі чотири зв'язки є типу "один до багатьох", окрім ROLE_APPLICATION – користувач може надсилати декілька заявок на певну роль одночасно, для прикладу він може надіслати заявку на роль вчителя та модератора одночасно.

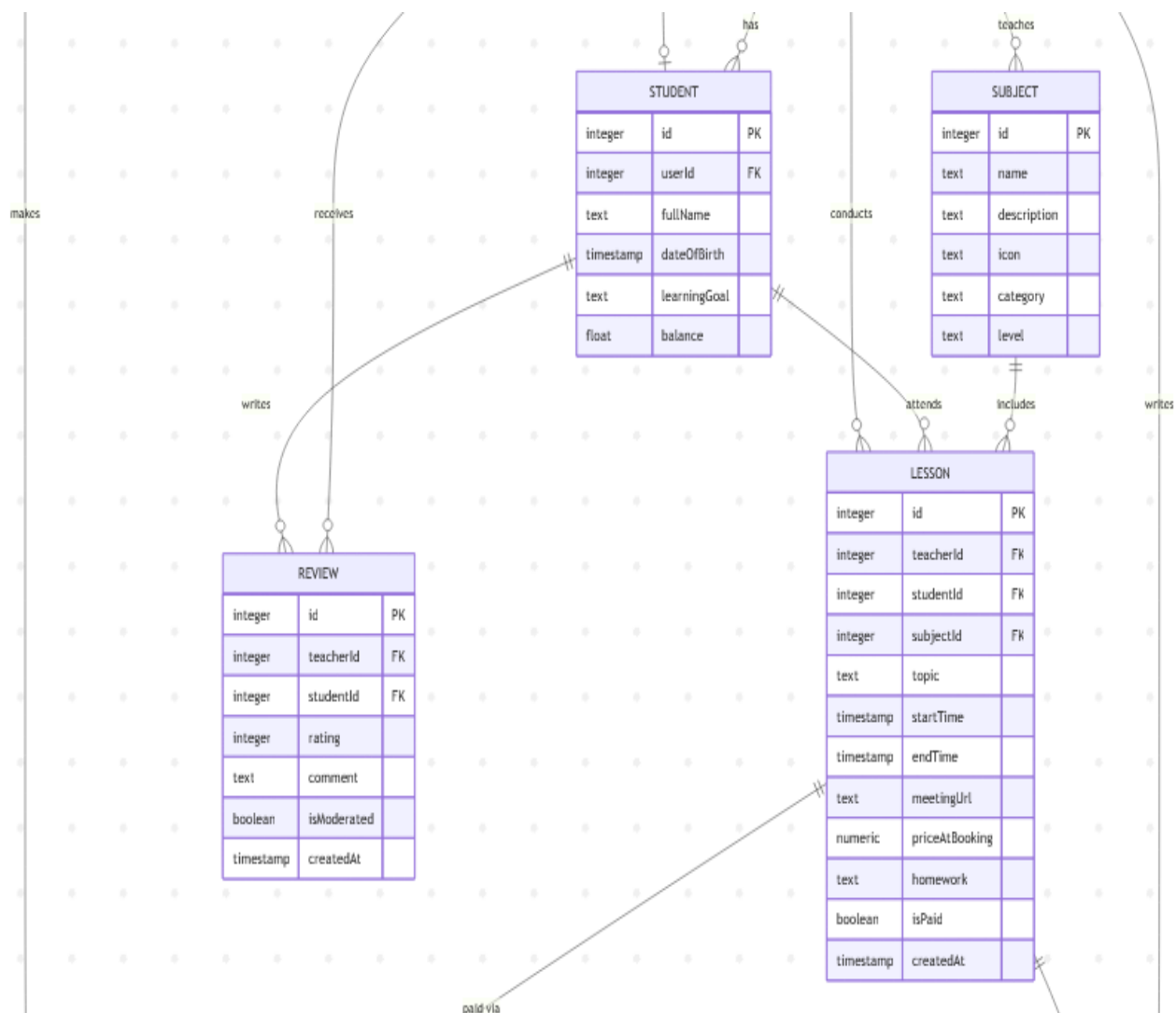


Рис. 2.4 – Друга частина діаграми класів

На цій частині діаграми зображено навчальне ядро системи з трьома ключовими сутностями, що сходяться до LESSON. STUDENT містить профіль учня, SUBJECT описує предмет із категорією та рівнем складності, а сама сутність LESSON є центральним агрегатором: вона зберігає зовнішні ключі до викладача, студента та предмета одночасно, а також усі деталі заняття - тему, час початку й кінця, посилання на зустріч, ціну на момент бронювання, домашнє завдання та статус оплати. REVIEW пов'язана зі студентом і викладачем, містить числовий рейтинг, коментар і прапорець модерації. Зв'язки відображують те, що один студент може відвідувати багато уроків, один предмет може входити до багатьох уроків, а кожен урок має рівно одного викладача, одного студента й один предмет.

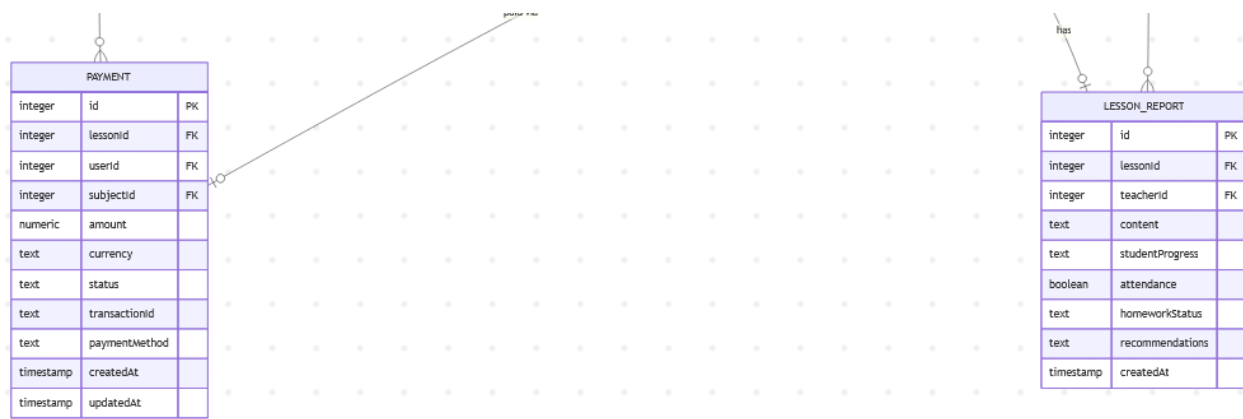


Рис. 2.5 - Третя частина діаграми класів

На цій частині діаграми зображено дві сутності (рис. 2.5), що обслуговують урок після його проведення. PAYMENT фіксує фінансову сторону: вона прив'язана до уроку, користувача та предмета через зовнішні ключі, зберігає суму, валюту, статус транзакції та метод оплати з часовими мітками створення й оновлення. LESSON_REPORT відповідає за педагогічну звітність: прив'язана до конкретного уроку та викладача, містить текстовий зміст заняття, оцінку прогресу студента, відмітку про відвідуваність, статус виконання домашнього завдання та рекомендації для подальшого навчання. Обидві сутності є результатом завершеного уроку - PAYMENT підтверджує його оплату, а LESSON_REPORT документує його якість і результати.

2.4. Проєктування та реалізація алгоритмів роботи системи

Алгоритмічне забезпечення відіграє фундаментальну роль у гарантуванні ефективності та логічної коректності функціонування освітньої платформи. Воно охоплює широкий спектр завдань: від валідації вхідних даних при автентифікації до складного управління фінансовими транзакціями та розкладом занять. Для формалізації логіки роботи сервісу та деталізації етапів взаємодії акторів із програмним продуктом було розроблено діаграми діяльності.

У контексті даного проєкту ключовим алгоритмом є процес доступу до функціональних можливостей платформи. Хоча система передбачає режим гостьового перегляду, повний спектр сервісів - зокрема запис на уроки, проведення оплати та ведення звітності - стає доступним лише після

ідентифікації суб'єкта. Для детального аналізу початкового етапу взаємодії було сформовано алгоритм, що відображує процеси логіну. Алгоритм охоплює всі кроки: від введення персональних даних до успішної генерації сесійних токенів і надання доступу до особистого кабінету. Такий підхід гарантує безпеку персональних даних та забезпечує розмежування прав доступу згідно з обраною рольовою моделлю. Користувач переходить на сторінку входу, де вводить свій email і пароль. Фронтенд виконує базову перевірку введених даних, а потім надсилає запит до бекенду. Сервер перевіряє існування користувача та правильність облікових даних через ORM. Якщо все вірно - бекенд повертає токен авторизації та дані користувача, які зберігаються на фронтенді для подальших запитів. У разі помилки користувачу показується повідомлення з відповідним статусом.

Для забезпечення високого рівня безпеки персональних даних та одночасно безперервного користувацького досвіду було реалізовано алгоритм механізму подвійних токенів. Короткоживучий Access-токен використовується фронтендом для авторизації кожного HTTP-запиту до захищених ендпоінтів бекенду. Коли термін дії Access-токена вичерпується, сервер повертає помилку зі статусом 401 [21]. Алгоритм на стороні клієнта за допомогою спеціального перехоплювача автоматично призупиняє виконання поточного запиту та надсилає прихований запит на бекенд із Refresh-токеном, який зберігається у локальному сховищі. Сервер валідує Refresh-токен, звіряє його із записом у базі даних, і в разі успіху генерує нову пару токенів. Клієнтська частина оновлює їх у пам'яті та повторно виконує початковий запит користувача, що робить процес оновлення сесії абсолютно непомітним для суб'єкта.

Невід'ємною складовою функціонування освітньої платформи є алгоритм бронювання занять та проведення оплати. Після вибору студентом доступного слоту в розкладі викладача, фронтенд ініціює процес створення резервування. Алгоритм передбачає звернення до інтегрованого платіжного шлюзу [22]. У разі успішного підтвердження транзакції сервером банку, бекенд системи валідує платіж, оновлює статус заняття у базі даних на "оплачено" та закріплює обраний

час за конкретним студентом. Усі ці зміни миттєво відображаються в особистих кабінетах як студента, так і викладача. У випадку відхилення платежу система автоматично скасовує попереднє бронювання та інформує користувача про помилку, пропонуючи повторити спробу

Ще одним критично важливим алгоритмом є процес подачі та затвердження заявок на отримання ролі викладача. Користувач заповнює спеціалізовану форму на клієнтській стороні, прикріплюючи необхідну інформацію щодо своєї кваліфікації. Бекенд приймає ці дані, проводить їх санітизацію та створює запис заявки в базі даних зі статусом "на розгляді". Адміністратор платформи через відповідний інтерфейс отримує доступ до переліку нових заявок. Алгоритм обробки дозволяє адміністратору ухвалити або відхилити заявку. У разі схвалення бекенд автоматично ініціює зміну рольової моделі користувача в системі, розширюючи його права доступу, та надсилає відповідне сповіщення.

Важливою алгоритмічною особливістю реалізації клієнтської частини є механізм управління станом та кешування серверних даних. Для уникнення надлишкових мережових запитів алгоритм роботи фронтенду передбачає інвалідацію кешу лише за умови успішного виконання мутацій (наприклад, після створення уроку або зміни статусу скарги). Це гарантує, що клієнтський застосунок завжди відображає актуальні дані з бази даних, мінімізуючи при цьому навантаження на сервер та зменшуючи час очікування для кінцевого користувача.

Окремої уваги заслуговує алгоритм пошуку та фільтрації спеціалістів, який є базовим інструментом навігації на платформі. Особливістю даної реалізації є перенесення логіки фільтрації та сортування безпосередньо на клієнтську сторону. Після отримання базового масиву даних про викладачів від сервера, клієнтський застосунок бере на себе обробку критеріїв пошуку. Коли користувач обирає певну категорію, предмет, бажаний ціновий діапазон або змінює параметри сортування за рейтингом, алгоритм на фронтенді миттєво застосовує ці параметри до наявного пулу даних. Такий підхід дозволяє динамічно

оновлювати список профілів викладачів на екрані без необхідності надсилання додаткових мережевих запитів до бази даних при кожній зміні фільтрів. Це суттєво знижує обчислювальне навантаження на бекенд та забезпечує максимально швидкий відгук інтерфейсу, покращуючи загальний користувацький досвід.

Крім того, для забезпечення високого рівня комунікації між учасниками платформи та оперативного інформування про хід освітнього процесу, розроблено алгоритм подієво-орієнтованих системних сповіщень. Логіка цього алгоритму базується на відстеженні змін стану сутності запланованого заняття. Якщо викладач ініціює будь-яку зміну параметрів уроку - наприклад, коригує час його проведення, оновлює деталі або скасовує зустріч - серверна частина під час виконання цього запиту автоматично перехоплює подію оновлення. Наслідком цієї події є миттєва генерація та відправка відповідного транзакційного сповіщення учню, який закріплений за даним заняттям. Такий реактивний підхід гарантує, що студенти завжди володіють найактуальнішою інформацією щодо своїх занять безпосередньо від викладачів, що мінімізує організаційні непорозуміння та підвищує надійність роботи сервісу.

Логічним завершенням циклу взаємодії між учнем та викладачем є алгоритм формування та опрацювання системи відгуків, що відіграє ключову роль у підтримці актуальності внутрішнього рейтингу спеціалістів. Після успішного підтвердження заявки учня на навчання у викладача система автоматично відкриває для студента можливість залишити текстовий коментар та виставити числову оцінку за визначеною шкалою. Алгоритм обробки цієї дії передбачає сувору валідацію на сервері: перевіряється факт реального запису між цими користувачами для унеможливлення накрутки або спаму. Після успішного збереження нового відгуку в базі даних, бекенд ініціює фоновий процес перерахунку загального середнього рейтингу відповідного викладача. Оновлене значення агрегується та негайно синхронізується з публічним профілем спеціаліста. Це створює надійний цикл зворотного зв'язку: нові оцінки безперервно впливають на позицію викладача в загальному каталозі,

гарантуючи, що алгоритм сортування на клієнтській стороні завжди оперує об'єктивними та найсвіжішими показниками якості надання освітніх послуг.

2.5. Реалізація функціоналу системи

Першочерговим етапом розробки архітектури системи є проектування реляційної моделі бази даних. Для візуалізації структури даних та детермінації зв'язків між об'єктами доцільно використовувати графічні інструменти моделювання, такі як сервіс draw.io. Створення концептуальної ER-діаграми дозволяє наочно визначити склад таблиць, специфікувати типи полів та встановити логічні зв'язки різної складності:

- Один-до-одного - наприклад, зв'язок між базовим обліковим записом користувача та його спеціалізованим профілем учня чи вчителя;
- Один-до-багатьох - зв'язок між користувачем та його сесійними токенами або між вчителем та звітами про проведені уроки;
- Багато-до-багатьох - інтенсивна взаємодія між вчителями та предметами навчання або між учнями та викладачами у межах освітнього процесу.

На основі розробленої візуальної моделі здійснюється програмна реалізація бази даних із використанням TypeORM - сучасної об'єктно-реляційної бібліотеки для середовища NestJS. Вибір TypeORM зумовлений її повною підтримкою TypeScript, що гарантує високу типізацію та стабільність коду [17].

У межах реалізації кожна таблиця з проєкту перетворюється на відповідний клас-сутність. Використання декораторів дозволяє декларативно описувати структуру таблиць, типи даних та обмеження цілісності (наприклад, унікальність пошти чи обов'язковість заповнення імені).

Особлива увага приділяється програмному відображенню реляційних зв'язків. За допомогою декораторів @OneToOne(), @ManyToOne() та @ManyToMany() бібліотека TypeORM автоматично формує необхідні зовнішні ключі та допоміжні таблиці зв'язків у PostgreSQL. Такий підхід забезпечує синхронізацію між програмним кодом та фізичною структурою бази даних, дозволяючи розробнику маніпулювати складними об'єктами через інтерфейси

класів, що значно прискорює розробку та полегшує подальшу підтримку платформи [16]. Для наочної демонстрації підходу до ORM-маппінгу наведено фрагмент програмного коду класу-сутності Subject, який відповідає за збереження даних про навчальні дисципліни. Цей клас містить унікальний ідентифікатор, текстові поля для збереження назви предмета, його опису та графічної іконки, а також масив переліку рівнів складності навчання, до яких належить дисципліна. Крім того, архітектура сутності передбачає встановлення зв'язку «багато до багатьох» із профілями викладачів, що реалізується за допомогою спеціального декоратора та дозволяє гнучко закріплювати за кожним предметом декількох відповідних спеціалістів.

Як видно з нижче наведеного фрагменту коду, використання TypeScript-декораторів @Column дозволяє чітко типізувати поля бази даних, включаючи роботу з переліченнями та масивами. Декоратори @ManyToOne та @OneToMany декларативно описують зв'язки поточної сутності з профілями викладачів та записами про уроки, делегуючи створення зовнішніх ключів та проміжних таблиць підсистемі TypeORM.

```
@Entity()
export class Subject {
  @PrimaryGeneratedColumn()
  id!: number;
  @Column({ type: "text", unique: true })
  name: string = "";
  @Column({ type: "text" })
  description: string = "";
  @Column({ type: "text" })
  icon: string = "";
  @Column({
    type: "enum",
    enum: Levels,
    array: true,
  })
  level: Levels[] = [];
  @ManyToOne(() => TeacherProfile, (teacher) => teacher.subjects)
  teachers!: TeacherProfile[];
}
```

У якості основного сховища для медіафайлів у межах розробленої системи було обрано локальний фізичний носій сервера. Таке рішення є методологічно обґрунтованим для освітньої платформи, оскільки воно забезпечує оптимальний

баланс між швидкістю реалізації та економічною ефективністю проєкту. Використання дискового простору сервера дозволяє уникнути інтеграції з високовартісними хмарними сервісами, що є критично важливим аспектом при розробці та тестуванні академічних стартап-рішень.

З точки зору архітектури NestJS, використання локального сховища дозволяє максимально ефективно використовувати вбудовані можливості фреймворку, зокрема перехоплювачі та вбудовану підтримку бібліотеки Multer [24]. Це дозволяє реалізувати процес завантаження файлів як ізольований сервіс, що інтегрується в модульну структуру застосунку без необхідності налаштування складних мережових протоколів для взаємодії із зовнішніми хмарними API. Такий підхід спрощує механізм валідації типів файлів та їхнього розміру безпосередньо в контурі сервера застосунку.

Програмна реалізація збереження медіафайлів на рівні контролера здійснюється за допомогою перехоплювачів. На лістингу продемонстровано метод створення профілю викладача із завантаженням його фотографії.

У наведеному фрагменті коду інструмент FileInterceptor автоматично перехоплює бінарний файл із HTTP-запиту (Додаток В). Для уникнення конфліктів імен при одночасному завантаженні файлів різними користувачами, вбудована функція filename генерує для кожного зображення унікальний 32-значний шістнадцятковий хеш, зберігаючи оригінальне розширення файлу. Після успішного збереження на диск, сервісу бізнес-логіки передається сформований URL для запису в базу даних [23].

Вибір хмарного сховища Firebase для збереження медіаконтенту позитивно впливає на показники продуктивності системи [6]. Оскільки віддача файлів делегується спеціалізованій інфраструктурі Firebase, це суттєво знижує навантаження на основний сервер, де функціонує логіка NestJS. Завдяки використанню глобальної мережі доставки контенту, час доступу до даних мінімізується незалежно від розташування клієнта. Це гарантує стабільну швидкість відгуку інтерфейсу та високу пропускну здатність системи при

одночасних запитах від великої кількості користувачів, що є важливим параметром нефункціональних вимог.

Доцільність впровадження сучасних хмарних рішень в архітектуру освітньої платформи зумовлена необхідністю оптимізації процесів розгортання та забезпечення високої доступності без додаткових витрат на DevOps-адміністрування. Використання платформи Render для розміщення бекенду на базі NestJS дозволяє повністю абстрагуватися від ручного управління серверами та операційними системами [25]. Платформа автоматизує процеси збірки та доставки коду безпосередньо з репозиторію, гарантуючи ідентичність середовищ тестування та продакшену. Це нівелює проблему непередбачуваної поведінки системи та забезпечує стабільну роботу програмного продукту з коробки.

Своєю чергою, вибір Vercel та Neon як супутніх інфраструктурних компонентів базується на їхній високій ресурсній ефективності та сучасних підходах до масштабування. Vercel забезпечує миттєву доставку клієнтської частини платформи через глобальну мережу, мінімізуючи затримки для кінцевих користувачів. Інтеграція ж бессерверної бази даних Neon доповнює цю архітектуру: на відміну від класичних баз даних, які вимагають постійного резервування потужностей, Neon динамічно масштабує обчислювальні ресурси залежно від поточного навантаження [27]. Такий синергетичний підхід дозволяє суттєво знизити витрати на інфраструктуру, забезпечуючи при цьому високу продуктивність, відмовостійкість та готовність системи до швидкого зростання кількості користувачів.

Для гарантування безпеки персональних даних та розмежування прав доступу між учнями, викладачами та адміністраторами освітньої платформи було впроваджено систему автентифікації на основі стандарту JSON Web Token. Вибір даної технології зумовлений її безстатусною природою, що є критично важливим для сучасних вебзастосунків та мобільних аплікацій. На відміну від традиційної моделі сесій на основі ідентифікаторів у пам'яті сервера, JWT дозволяє передавати всю необхідну інформацію про користувача безпосередньо у самому токени. Це суттєво підвищує продуктивність системи, оскільки серверу

не потрібно щоразу звертатися до бази даних або кешу для перевірки активності сесії, що дозволяє легко масштабувати платформу при зростанні кількості одночасних навчальних занять.

Структурно кожен токен складається з трьох зашифрованих частин: заголовка, корисного навантаження та цифрового підпису. У корисному навантаженні зберігаються нечутливі дані користувача, такі як унікальний ідентифікатор суб'єкта, його роль у системі та термін дії токена. Ключовим елементом безпеки є саме третя частина - підпис, який створюється шляхом хешування перших двох частин із використанням секретного ключа, що зберігається виключно на сервері. Це гарантує цілісність даних: якщо зломисник спробує змінити роль користувача у токені, підпис автоматично стане недійсним, і сервер відхилить такий запит як скомпрометований.

На серверній частині застосунку, реалізованій на фреймворку NestJS, логіка авторизації інтегрована через спеціалізовані модулі та стратегії автентифікації. Після успішної перевірки пари логін-пароль, сервер ініціює генерацію двох типів токенів: короткострокового Access Token та довгострокового Refresh Token. Access Token використовується для ідентифікації кожного HTTP-запиту до захищених ресурсів API, тоді як Refresh Token слугує механізмом для отримання нової пари токенів без повторного введення облікових даних користувачем. Для підвищення рівня безпеки Refresh Token не просто видається клієнту, а його криптографічний хеш зберігається у базі даних, що дозволяє адміністрації системи миттєво відкликати доступ у разі підозрілої активності або втрати пристрою користувачем.

Контроль доступу на сервері забезпечується за допомогою вбудованих механізмів NestJS - перехоплювачів доступу. Кожен запит до ендпоінтів, що стосуються фінансових операцій, редагування профілів або створення звітів про уроки, проходить через JWT-парсинг. Вона автоматично витягує токен із заголовка авторизації, перевіряє його валідність та впроваджує розкодовані дані користувача безпосередньо в об'єкт запиту. Це дозволяє розробнику зручно маніпулювати правами доступу, використовуючи декоратори ролей, та гарантує,

що учень не зможе отримати доступ до журналів викладача або адміністративної панелі керування платежами.

Клієнтська частина платформи, розроблена на React відповідає за життєвий цикл токенів та їх безпечне збереження. При отриманні токенів від сервера, вони розміщуються у захищених сховищах - використовуються LocalStorage. Основним інструментом взаємодії з API є перехоплювачі бібліотеки Axios, які автоматично додають актуальний токен до кожного запиту. Це дозволяє повністю абстрагувати логіку безпеки від компонентів інтерфейсу, роблячи процес автентифікації прозорим для розробника та кінцевого користувача.

Важливою частиною клієнтської логіки є реалізація механізму тихого оновлення сесії. У випадку, коли термін дії Access Token вичерпується, сервер повертає помилку з відповідним кодом доступу. Клієнтський перехоплювач автоматично відловлює цей стан, призупиняє всі вихідні запити та ініціює запит на оновлення через Refresh Token. У разі успішного оновлення, нова пара токенів зберігається у сховищі, а всі призупинені запити повторно надсилаються до сервера з уже новим токеном. Такий підхід забезпечує безперервність користувацького досвіду, дозволяючи студенту продовжувати урок або переглядати матеріали без раптових переривань на повторну авторизацію.

Фінальний рівень захисту JWT-авторизації базується на суворому дотриманні політик безпеки при передачі даних та зберіганні конфігурацій [7]. Усі секретні ключі, що використовуються для підпису токенів, а також параметри їхнього часу життя, винесені у змінні оточення, що унеможлиблює їх потрапляння у відкритий доступ через систему контролю версій. Крім того, обов'язковою умовою функціонування платформи є використання протоколу шифрування TLS [30]. Це гарантує, що токени, які передаються у заголовках запитів, не будуть перехоплені зловмисниками в процесі передачі через мережу, забезпечуючи надійний захист освітнього середовища від несанкціонованого втручання.

Фундаментом розробленої системи є розгалужена модель управління обліковими записами, яка забезпечує гнучку та безпечну взаємодію між учнями та викладачами. Оскільки освітній процес вимагає високої довіри до кваліфікації спеціалістів, ключовим етапом розробки стало створення модуля верифікації викладачів. На відміну від звичайних користувачів, акаунти вчителів проходять через цикл модерації у межах сутності `role_application`.

На клієнтській стороні взаємодія із сервером та кешування результатів реалізовано за допомогою хука `useQuery`.

Нижче наведено фрагмент компонента відображення профілю викладача [29]. Цей програмний код демонструє декларативний підхід до управління станом даних. Замість ручного використання ефектів та станів, бібліотека `TanStack Query` автоматично обробляє процес завантаження та перехоплює можливі мережеві помилки. Параметр `queryKey` слугує унікальним ідентифікатором для кешу, що дозволяє миттєво відображати дані при повторному відвідуванні сторінки без зайвих звернень до API.

```
export default function TeacherPage() {
  const { id } = useParams();
  const { isLoading, data, isError, error } = useQuery({
    queryKey: ["findTeacherById", id],
    queryFn: () => findTeacherById(+id!),
    enabled: !!id
  });
  if (isError) return <ErrorMessage error={error} />;
  if (isLoading) return <Spinner />;
  return (
    <div className="heroInfo">
      <div className="heroName">{data?.fullName}</div>
      { /* Відображення іншої інформації профілю */ }
    </div>
  );
}
```

Центральне місце в управлінні акаунтом займає функціонал оновлення персональних даних, що вимагає чіткої синхронізації між фронтендом, бізнес-логікою сервера та реляційним сховищем. Процес оновлення профілю учня або викладача побудований на принципах суворої валідації вхідних даних для збереження цілісності системи. Схематично послідовність взаємодії між компонентами системи під час редагування профілю відображено на рисунку 2.6.

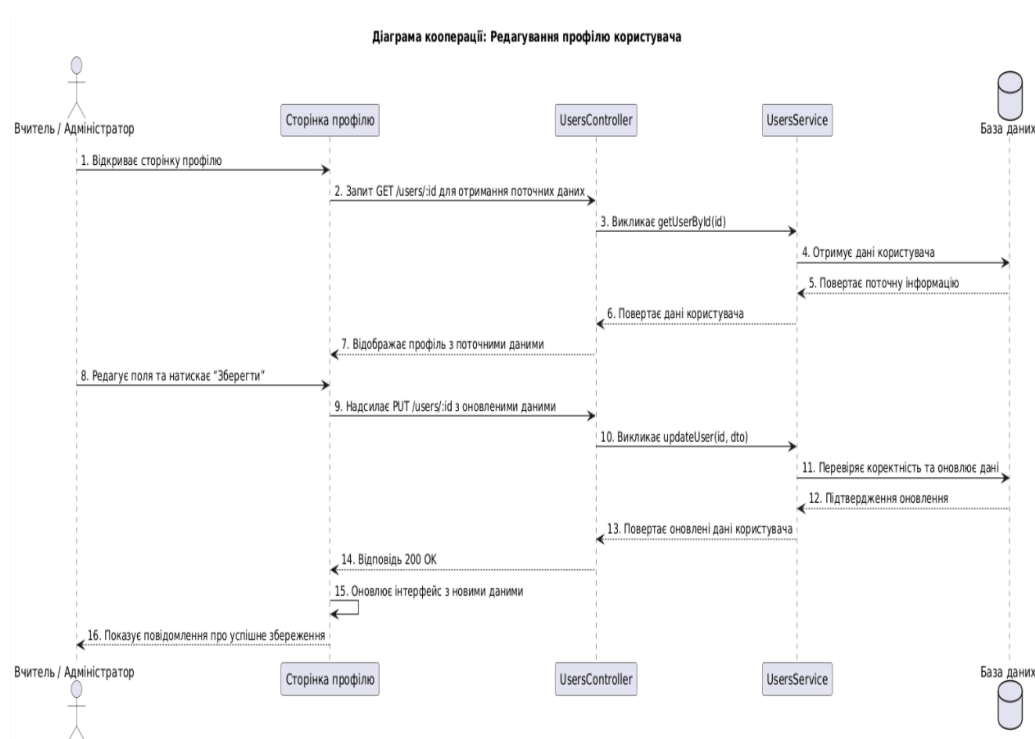


Рис. 2.6 - Діаграма послідовності оновлення акаунта учня

Як демонструє наведена вище діаграма (рис. 2.6), процес розпочинається з ініціації запиту на отримання поточних даних профілю. UsersController через сервісний шар звертається до бази даних, агрегуючи інформацію з таблиць user та відповідного профільного розширення. Після внесення змін користувачем, фронтенд надсилає оновлений об'єкт на сервер, де відбувається повторна перевірка на відповідність типам даних та бізнес-правилам системи. Після успішної фіксації змін у PostgreSQL, сервер надсилає підтвердження, що ініціює автоматичне оновлення кешу на клієнті через TanStack Query. Такий архітектурний підхід дозволяє підтримувати високу актуальність професійної інформації про викладачів, що є критичним для коректного функціонування системи бронювання та пошуку.

Висновки до другого розділу

Проектування архітектури освітнього сервісу дозволило сформувати цілісну екосистему, де пріоритетом є безперебійна взаємодія між викладачем та учнем. Аналіз вимог до платформи став підґрунтям для розробки гнучкої рольової моделі, яка враховує специфічні потреби кожного актора: від модерації

професійних анкет адміністратором до оперативного бронювання занять студентами. Спроектowana система орієнтована на високу відмовостійкість та здатність до горизонтального масштабування, що є критичним для сервісів із високою інтенсивністю запитів у реальному часі.

Логічна структура застосунку пройшла етап верифікації через розробку UML-моделей. Зокрема, діаграми активностей дозволили деталізувати складні алгоритми верифікації викладачів та обробки фінансових транзакцій, а побудована діаграма класів чітко розмежувала зони відповідальності програмних модулів. Це забезпечило прозорість архітектури та спростило подальшу імплементацію бізнес-логіки на базі NestJS.

Технологічний стек системи базується на поєднанні реляційної надійності PostgreSQL та сучасних стандартів безпеки. Розроблена схема бази даних через TypeORM гарантує цілісність інформації про навчальні плани та історію оплат. Безпека користувацьких даних забезпечується гібридною моделлю авторизації з використанням пари JWT-токенів, що дозволяє підтримувати довготривалі сесії без ризику для конфіденційності облікових записів.

Особлива увага в проєкті приділена мінімізації латентності інтерфейсу. Завдяки впровадженню серверного кешування на базі Redis та інтелектуальному управлінню станом на стороні клієнта через TanStack Query, вдалося досягти високої швидкодії при виконанні ресурсомістких операцій пошуку та фільтрації. Механізм оптимістичних оновлень та посторінкова навігація дозволили створити динамічний інтерфейс, який нівелює затримки мережевої взаємодії та забезпечує комфортний користувацький досвід на будь-яких типах пристроїв.

Сформована в ході проєктування технічна документація та візуальні моделі є вичерпною базою для переходу до етапу програмної реалізації, гарантуючи відповідність продукту сучасним вимогам до продуктивності та захищеності інформаційних систем.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СЕРВІСОМ

3.1. Порядок встановлення та налаштування параметрів системи

Інфраструктура розробленої вебплатформи онлайн-школи базується на децентралізованій хмарній архітектурі, що поєднує концепції «платформа як послуга» та безсерверних обчислень [19]. Такий підхід дозволяє повністю делегувати адміністрування фізичного обладнання хмарним провайдерам, забезпечуючи високу доступність навчальних ресурсів, автоматичне масштабування під час пікових навантажень та безперервну інтеграцію [20]. Взаємодія між основними вузлами системи відображена на діаграмі розгортання (рис. 3.1).

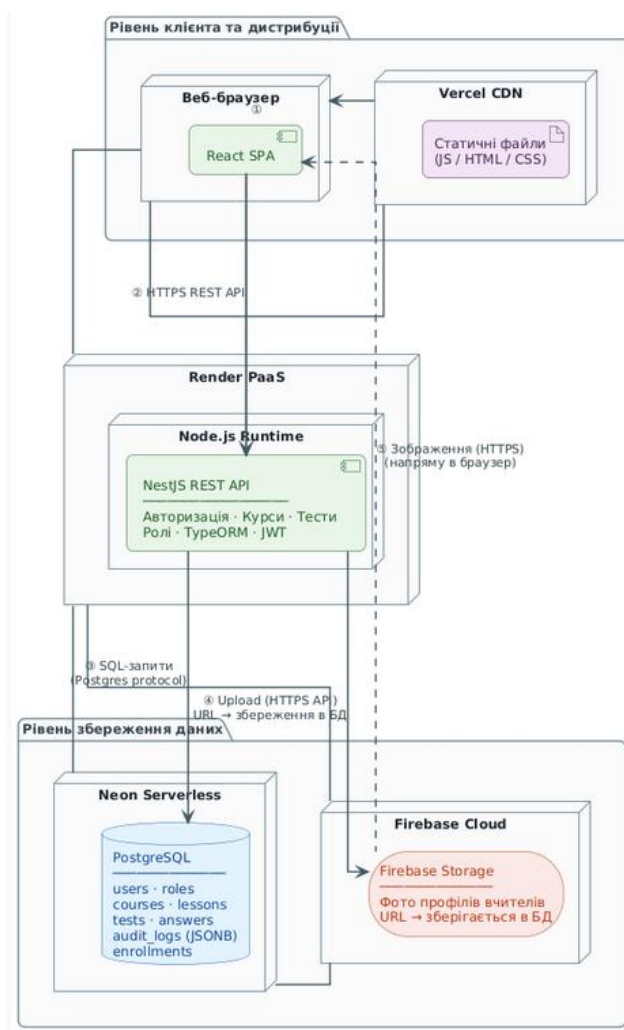


Рис. 3.1 - Діаграма розгортання вебплатформи онлайн-школи

Важливою складовою надійності представленої архітектури є організація безпечної взаємодії між її розподіленими компонентами. Обмін даними між веббраузером клієнта та сервером здійснюється виключно зашифрованими

каналами зв'язку. Оскільки сервер функціонує в "stateless" середовищі, для управління сесіями та підтвердження повноважень користувачів імплементовано стандарт JSON Web Token. Такий підхід дозволяє бекенду автономно та швидко верифікувати права доступу до захищених маршрутів, уникаючи надлишкових звернень до бази даних.

Клієнтська частина платформи, яка виступає основним інтерфейсом взаємодії для студентів та викладачів, реалізована за допомогою бібліотеки React та розміщена в хмарній екосистемі Vercel. Дане рішення автоматично оптимізує фронтенд-застосунок і здійснює його дистрибуцію через глобальну мережу доставки контенту. Це мінімізує затримки під час завантаження сторінок курсів, перегляду уроків чи взаємодії з особистим кабінетом, незалежно від географічного розташування користувача [26].

Обробка бізнес-логіки та взаємодія з даними покладена на серверну частину, створену на базі платформи NestJS та розгорнуту на хостинг-платформі Render. Сервер функціонує в ізольованому PaaS-середовищі, приймаючи та обробляючи виключно стандартні HTTP/HTTPS-запити від клієнта через REST API [28]. До зони відповідальності бекенду належать процеси автентифікації користувачів, розмежування прав доступу, керування структурою навчальних курсів, фіксація прогресу навчання та генерація звітів.

Для надійного зберігання структурованої інформації інтегровано безсерверне реляційне сховище Neon на базі СУБД PostgreSQL. Ключовою технологічною перевагою Neon є архітектурний поділ обчислювальних ресурсів та збережених даних. Це дозволяє базі даних гнучко адаптуватися до навантаження і автоматично зупиняти обчислювальні вузли у періоди низької активності, оптимізуючи витрати ресурсів.

Окрім того, використання платформи Neon ефективно вирішує проблему керування пулом з'єднань, яка є критичною для хмарних архітектур з високою конкурентністю запитів. Завдяки вбудованим механізмам оптимізації, база даних здатна стабільно обробляти велику кількість одночасних транзакцій від сервера NestJS без деградації продуктивності. Ця характеристика є надзвичайно

важливою для освітньої системи під час пікових навантажень -наприклад, у періоди масового проходження тестів або одночасного старту онлайн-уроків у різних групах. У таких сценаріях бессерверна архітектура забезпечує миттєве автомасштабування обчислювальних потужностей для покриття потреб трафіку, гарантуючи високу доступність даних без необхідності ручного втручання чи попереднього резервування серверів.

Робота з графічними матеріалами платформи носить локалізований характер і обмежується збереженням фотографій профілів викладацького складу. Для надійного збереження цих зображень інтегровано хмарне об'єктне сховище Firebase Storage. Процес завантаження графічних файлів координується серверною частиною, яка після успішної автентифікації та передачі знімка у хмару генерує унікальну публічну URL-адресу. Отримане посилання інкапсулюється безпосередньо в базу даних PostgreSQL як атрибут сутності профілю вчителя, що дає змогу оперативно рендерити фотокартки викладачів на клієнтській стороні під час завантаження інформації про курс чи персональну сторінку користувача.

Окремо варто відзначити механізм отримання медіаконтенту кінцевим користувачем. Як відображено на діаграмі розгортання, безпосереднє завантаження статичних зображень для відображення в інтерфейсі виконується клієнтським застосунком безпосередньо з Firebase Storage. Делегування віддачі медіафайлів спеціалізованому хмарному сховищу повністю знімає це навантаження з основного вузла NestJS. Завдяки такій мікросервісній взаємодії економиться пропускна здатність бекенду, а обчислювальні потужності Render спрямовуються виключно на виконання складних SQL-запитів до Neon та обробку бізнес-правил платформи.

У підсумку, обрана конфігурація керованих хмарних сервісів формує гнучку, відмовостійку та безпечну архітектуру вебплатформи. Відсутність потреби в утриманні власного локального сервера знижує ризики апаратних збоїв та гарантує стабільний доступ до навчального процесу в режимі 24/7.

3.2. Структура інтерфейсу вебзастосунку

Успіх взаємодії користувача з вебплатформою безпосередньо залежить від продуманості її візуального та функціонального простору. Презентована архітектура екранів освітньої екосистеми спроектована на основі сучасних принципів UI/UX-проектування, де особлива увага приділена стилістиці необруталізму. Цей напрямок вебдизайну є свідомим кроком назустріч максимальній доступності та інтуїтивності інтерфейсу. Такий підхід передбачає відмову від звичних градієнтів, складних текстур чи розмитих форм на користь чітких силуетів: елементи мають виразне чорне обведення та монолітні жорсткі тіні, що додає цифровому середовищу відчуття фізичного об'єму та структурної стабільності. Завдяки високому контрасту меж, користувач без зайвих зусиль ідентифікує інтерактивні зони. Колористична концепція побудована на гармонійному поєднанні лавандового відтінку, який психологічно асоціюється зі спокоєм та концентрацією, яскравого жовтого кольору, що стимулює до активності, та чистого білого тла для оптимізації читабельності контенту. Це не лише виділяє платформу з-поміж класичних, консервативних академічних сайтів, а й допомагає природно фокусувати зір на ключових інформаційних блоках. Інтерактивність інтерфейсу суттєво підсилюється мікроанімаціями - картки та графічні елементи нахиляються під невеликим кутом або зсуваються під час руху курсора, створюючи живий, тактильний відгук на дії відвідувача, що значно покращує загальний рівень залученості.

Знайомство з ресурсом починається з головного вітального простору, який виконує роль візитної картки проєкту та задає тон усьому подальшому процесу навчання (рис. 3.1). Ця зона розроблена з урахуванням стилістичних патернів сприйняття візуальної інформації, щоб моментально залучити гостя та пояснити цінність освітнього продукту в перші секунди перебування на сайті. Ліворуч, у зоні первинної уваги, розташовано масивне текстове звернення разом із лаконічним роз'ясненням місії школи, а праворуч - тематична ілюстрація, яка візуально підкріплює текстовий меседж. Тут же розміщено головний інструмент конверсії - акцентну кнопку із закликом до дії. Її стратегічне розташування та

контрастне виділення миттєво спрямовують потенційного учня до повного переліку доступних дисциплін, мінімізуючи кількість кроків від входу на сайт до початку навчання. Щоб зробити перший екран більш динамічним та сучасним, для нього налаштовано ефект плавної появи: під час відкриття сторінки весь блок синхронно піднімається знизу, поступово набуваючи повної прозорості. Таке анімаційне рішення не лише додає естетичної привабливості, але й плавно фокусує увагу користувача на центральному повідомленні, знижуючи показник відмов на етапі першого знайомства з платформою.

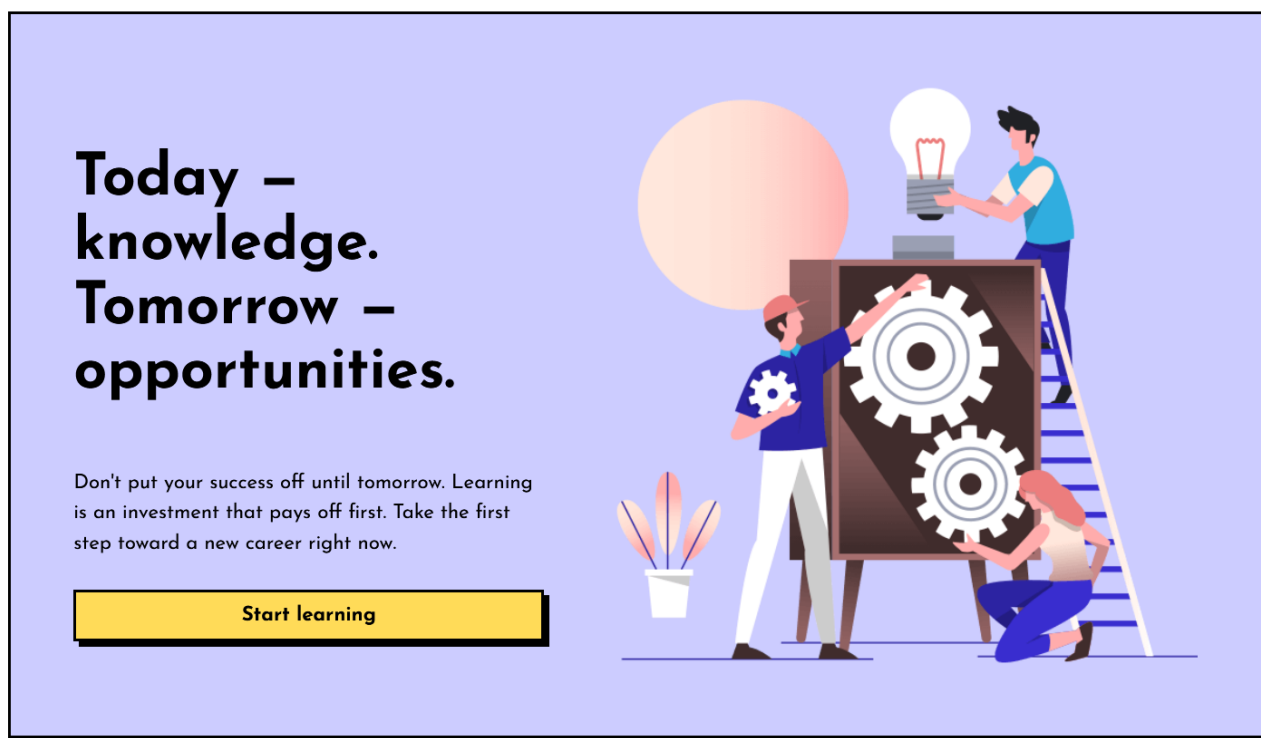


Рис. 3.1 - Банер головної сторінки

Нижче по вертикалі інтегровано навігаційний вузол, призначений для швидкого сортування навчальних напрямків (рис. 3.2). Система автоматично робить запит до бази даних і формує сітку з доступних категорій. Кожен освітній вектор оформлено у вигляді окремого графічного блоку, який візуально трансформується при наведенні, підказуючи користувачеві можливість кліку. Перехід за таким посиланням одразу відкриває відфільтровану сторінку з відповідними навчальними програмами, що значно скорочує шлях клієнта до старту занять.

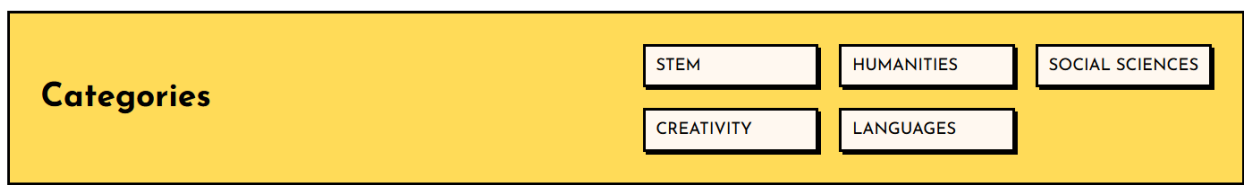


Рис. 3.2 -Розподілення категорій.

Для формування високого рівня довіри та демонстрації експертності нашої команди на сторінці виділено окремий інтерактивний модуль, присвячений провідному викладачеві платформи (рис. 3.3). Розумний алгоритм обирає найкращого спеціаліста на основі реальних оцінок студентів і виводить його картку. Візуальна структура презентує портрет спеціаліста, його повне ім'я та коротке професійне кредо. Важливі аналітичні показники - рівень задоволеності клієнтів, вартість однієї години лекції та тривалість викладацького стажу - винесені на окремі контрастні плашки. Прямо з цього блоку користувач може перейти до детального вивчення профілю репетитора та бронювання уроку.

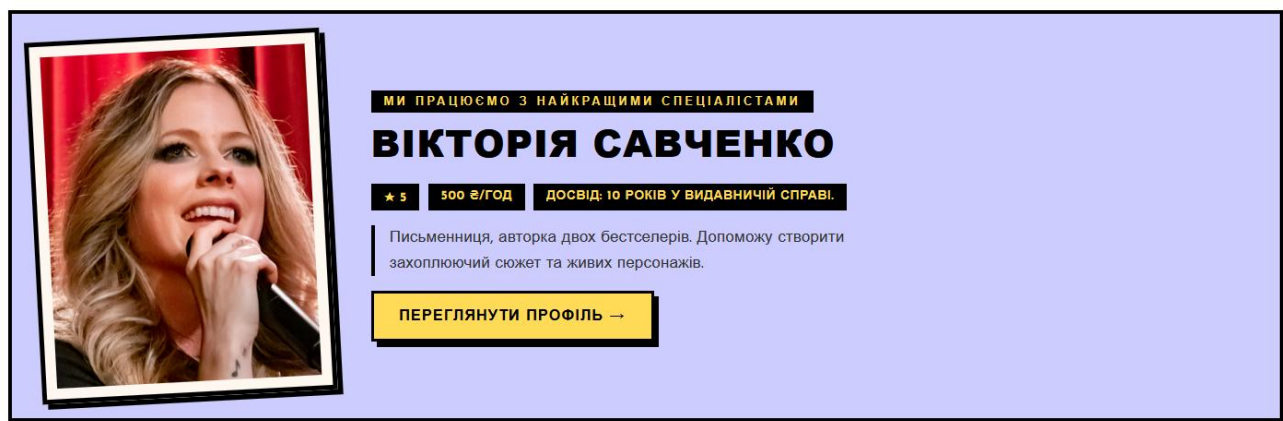


Рис. 3.3 - Блок із найбільш оціненим спеціалістом

Підсумковим елементом сторінки виступає фіксує блок, який виконує роль емоційного тригера (рис. 3. 4). Він транслює надихаючі гасла та філософію онлайн-школи, підштовхуючи відвідувача до прийняття рішення. Центральним елементом тут виступає масштабна кнопка реєстрації, натискання на яку відкриває форму створення облікового запису або авторизації. Це дозволяє логічно завершити сценарій ознайомчого перегляду та перевести відвідувача у статус активного студента.

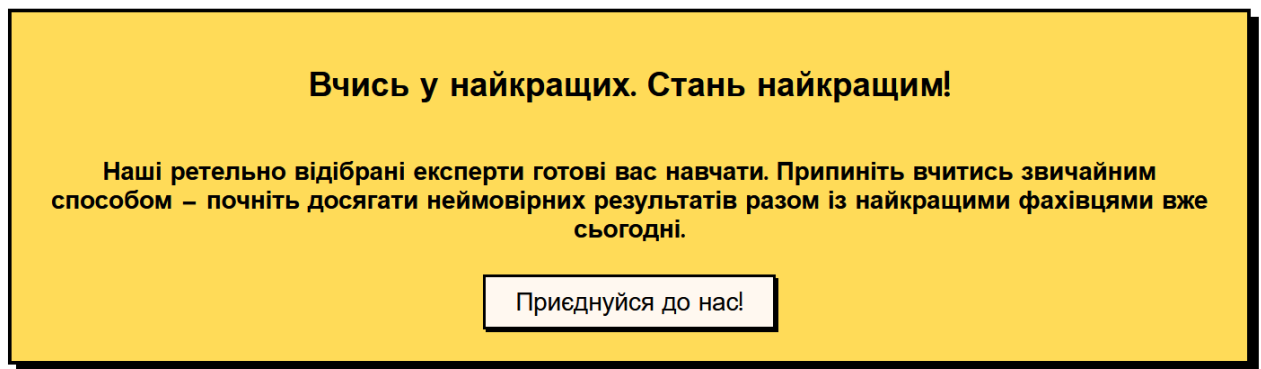


Рис. 3.4 - Блок із заохоченнями

Верхню частину інтерфейсу займає закріплена навігаційна панель (рис. 3.5), яка слугує основним орієнтиром для користувача під час вебсерфінгу і залишається видимою при прокручуванні сторінки. Її візуальне виконання повністю успадковує загальну концепцію необруталізму: використання кремового тла, масивного чорного контуру завтовшки 4 пікселі та жорсткої плоскої тіні. Фірмовий логотип «APOLLO SCHOOL» та інтерактивні пункти меню, оформлені шрифтом "Josefin Sans", реагують на наведення курсору легким зміщенням по діагоналі та забарвленням у пастельно-жовтий колір. Для десктопних екранів передбачено інтерактивні випадаючі списки, які з'являються за допомогою CSS-анімації з ефектом м'якого зсуву вниз та зміни прозорості, що спрощує доступ до внутрішніх сторінок сайту без додаткових перезавантажень.



Рис. 3.5 - Хедер

Функціональна логіка панелі безпосередньо інтегрована з глобальним сховищем стану автентифікації та динамічно адаптується під рольову модель доступу. Залежно від дозволів авторизованого профілю інтерфейс трансформує свій вміст: неавторизованим гостям пропонуються кнопки швидкого входу та перемикач мовних локалей, тоді як для студентів і викладачів відкриваються персоналізовані меню для керування профілем, графіком занять та поточними заявками. Для адміністративних ролей передбачено розширене випадаюче меню для оперативного доступу до модерації відгуків, керування контентом та перегляду системних логів. При зменшенні ширини екрана горизонтальна

навігація за допомогою медіа-запитів автоматично ховається, передаючи керування компактній кнопці-гамбургеру, яка розгортає вертикальний стек елементів із розширеною площею кліку під сенсорні дисплеї.

Нижню частину сторінки замикає компонент Footer (рис. 3.6), який містить модуль підписки на розсилку із фіктивною формою, де стандартне відправлення заблоковано через `preventDefault()`, та динамічну карту сайту, що адаптується під рольову модель користувача. Завдяки інтеграції з глобальним сховищем стану, підвал автоматично формує навігаційні блоки: авторизовані студенти та викладачі отримують швидкий доступ до внутрішніх кабінетів і списків занять, адміністратори - до інструментів додавання дисциплін, а гості - до інтерфейсів авторизації, що відображаються поряд зі сталими розділами юридичної та корпоративної інформації. Завершує структуру нижня плашка із захищеними зовнішніми посиланнями на соціальні мережі та динамічно обчислюваним поточним роком у полі копірайту.



Рис. 3.6 - Футер

Перейшовши на сторінку контактів, ми відразу взаємодіємо з інтерактивною картою (рис. 3.7), побудованою на базі OpenStreetMap: по ній можна вільно переміщуватися за допомогою миші, проте масштабування коліщатком навмисно заблоковано, щоб запобігти випадковому перехопленню прокрутки всієї сторінки [14]. Клікнувши на фіксований маркер, ми розгортаємо спливаючу підказку, текст якої автоматично підлаштовується під поточну мову інтерфейсу. Одразу під картою ми бачимо структуровану інформаційну таблицю

ЕЛЕКТРОННА ПОШТА	ipz222_zyeyu@student.ztu.edu.ua
АДРЕСА	Zhytomyr, Chudnivska 103
МОБІЛЬНИЙ ТЕЛЕФОН	+380688473823
РОБОЧІ ГОДИНИ	09:00 - 17:00
TELEGRAM	@Lalkazaurus

Коли ми відкриваємо сторінку реєстрації (рис. 3.8) та починаємо заповнювати поля електронної пошти, пароля, номера телефону та імені користувача, клієнтська валідація завдяки режиму `onChange` миттєво реагує на кожен натиск клавіші, виводячи підказки у разі невідповідності регулярним виразам чи обмеженням щодо довжини тексту. У нас є можливість в один клік повністю очистити форму за допомогою кнопки скидання або ж відправити її на сервер; під час обробки запиту хуком `useMutation` усі поля та кнопки автоматично блокуються, змінюючи стан головної кнопки на процес завантаження, що унеможливорює повторне відправлення дублюючих даних. Крім класичної реєстрації, ми можемо скористатися альтернативним швидким

входом через сторонній сервіс за допомогою кнопки Google. Якщо створення профілю завершується успішно, система автоматично записує наші дані та токен до глобального стейту, повністю очищує кеш попередніх запитів і перенаправляє нас на головну сторінку платформи, а у випадку виникнення мережових чи логічних збоїв - над полями вводу миттєво з'являється динамічне повідомлення про помилку сервера.

РЕЄСТРАЦІЯ

ВВЕДІТЬ ВАШУ ЕЛЕКТРОННУ ПОШТУ

ВВЕДІТЬ ВАШ ПАРОЛЬ

ВВЕДІТЬ НОМЕР ВАШОГО МОБІЛЬНОГО ТЕЛЕФОНУ

ВВЕДІТЬ ВАШЕ ІМ'Я КОРИСТУВАЧА

О Н О В И Т И В І Д П Р А В И Т И

А Б О

 ЗАРЕЄСТРУВАТИСЬ ЧЕРЕЗ GOOGLE

ВЖЕ МАЄТЕ АКАУНТ? [ВХІД](#)

Рис. 3.8 - Сторінка реєстрації

Коли ми переходимо до форми авторизації (рис. 3.9), інтерфейс пропонує нам ввести логін та пароль, здійснюючи "on-change" валідацію електронної пошти на відповідність коректному регулярному виразу безпосередньо в процесі введення символів. Під час натискання кнопки підтвердження активується хук useMutation, який відправляє дані на сервер і тимчасово блокує всі інтерактивні елементи, запобігаючи спам-запитам, а у разі невірної введення реквізитів над

кнопками миттєво рендериться локалізоване повідомлення про помилку. Як альтернативу, ми можемо скористатися кнопкою швидкої автентифікації через Google-акаунт або скинути введені дані відповідною кнопкою очищення. Якщо ж ми забули пароль або ще не маємо облікового запису, нижня частина форми пропонує додаткові навігаційні посилання для відновлення доступу та переходу на сторінку реєстрації, а при успішному вході система записує користувача в Zustand-стейт, очищує кеш запитів queryClient і перенаправляє нас на головну сторінку [18]. У випадку успішної реєстрації на електронну скриньку, вказану користувачем, приходить відповідний вітальний лист.

ВХОДЖЕННЯ В АКАУНТ

ВВЕДІТЬ ВАШУ ЕЛЕКТРОННУ ПОШТУ

мвфі

× Ви повинні ввести саме електронну скриньку

ВВЕДІТЬ ВАШ ПАРОЛЬ

.....

О Н О В И Т И В І Д П Р А В И Т И

А Б О

У В І Й Т И Ч Е Р Е З G O O G L E

Н Е М А Є А К А У Н Т У ? З А Р Е Є С Т Р У В А Т И С Ь

З А Б У Л И П А Р О Л Ь ?

Рис. 3.9 - Форма входу з відповідною помилкою валідації

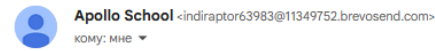
Коли ми переходимо за посиланням для відновлення пароля, перед нами відкривається лаконічна форма (рис. 3.10), де в процесі введення електронної пошти також активується миттєва "on-change" валідація на відповідність синтаксису email-адреси. Під час натискання кнопки надсилання хук useMutation блокує інпут та елементи керування, щоб запобігти повторним запитам, а у разі виникнення помилки на сервері інтерфейс одразу виводить локалізоване повідомлення про збій. Якщо ж сервер успішно обробляє наш email, компонент

повністю перебудовує свій вміст і замість форми відображає успішний екран із повідомленням про відправлене посилання для скидання пароля та зручною кнопкою для швидкого повернення на сторінку авторизації.

The screenshot shows a web form for password reset. At the top, the title "ВІДНОВЛЕННЯ ПАРОЛЯ" (Password Reset) is displayed. Below it is a text input field labeled "EMAIL" containing the placeholder text "your@email.com". At the bottom of the form are two buttons: "НАЗАД" (Back) and "НАДІСЛАТИ ПОСИЛАННЯ" (Send Link).

Рис. 3.10 - Форма відновлення паролю

Після надсилання відповідної форми на зареєстрований акаунт користувача приходить лист з посиланням на оновлення паролю (рис. 3.11).



The screenshot shows an email for password reset. The subject line is "RESET YOUR PASSWORD". The body of the email starts with "Hello, Lalkazaurus!". It then says: "We received a request to reset your password. Click the button below to set a new one:". Below this text is a yellow button labeled "RESET PASSWORD". Further down, a light yellow box contains the text: "This link is valid for 1 hour. If you didn't request a password reset, simply ignore this email — your account is safe." At the very bottom of the email, it says: "This is an automated message. Please do not reply to this email."

Рис. 3.11 - Надісланий на пошту лист для оновлення паролю

Коли ми переходимо за отриманим у листі посиланням для встановлення нового пароля, сторінка насамперед перевіряє наявність спеціального токена в параметрах URL: якщо він відсутній, інтерфейс одразу блокує доступ і відображає повідомлення про недійсне посилання з пропозицією запросити нове.

У разі успішної валідації токена відкривається форма (рис. 3.12), де ми вводим новий пароль та підтверджуємо його; при цьому вбудована функція watch у режимі реального часу звіряє обидва поля, миттєво виводячи попередження у разі їхньої невідповідності. Під час надсилання форми хук useMutation передає токен і новий пароль на сервер, традиційно блокуючи елементи введення від повторних кліків, а у випадку помилки рендерить відповідне сповіщення про збій. Якщо ж пароль оновлено успішно, компонент перемикається на екран успіху з інформаційним повідомленням і автоматично, за допомогою функції setTimeout із затримкою у 2,5 секунди, перенаправляє нас на сторінку авторизації для входу в систему з новими реквізитами.

Рис. 3.12 - Форма оновлення паролю

Коли ми заходимо на сторінку профілю (рис. 3.13), система насамперед перевіряє наш статус автентифікації: якщо ми не авторизовані, нас миттєво й примусово перенаправляє на сторінку входу. Для авторизованого ж користувача розгортається персональний профіль, де у верхній плашці одразу відображається ім'я та всі присвоєні ролі. Нижче, у структурованій сітці карток, ми бачимо детальні відомості про свій акаунт: електронну пошту, телефонний номер (або прочерк, якщо він відсутній), поточний статус системи та локалізовану дату останнього візиту. В залежності від наших активних ролей (студент чи

викладач), вбудований компонент дочірніх посилань динамічно підлаштовується і виводить спеціальні кнопки для миттєвого переходу у профільні навчальні кабінети, а знизу сторінки нам доступне окреме посилання для швидкого переходу до форми зміни пароля.

Рис. 3.13 - Сторінка профілю

Коли ми потрапляємо на сторінку зміни пароля, перед нами з'являється форма (рис. 3.14), де в процесі заповнення текстових полів інтерактивно відпрацьовує "on-change" валідація, яка стежить за обов'язковістю заповнення, перевіряє новий пароль на відповідність критеріям складності, а також за допомогою методу watch контролює, щоб новий пароль не збігався з поточним і водночас збігався з повторно введеним у нижньому полі. Ми маємо можливість скинути всі значення натисканням кнопки очищення або ініціювати збереження змін; під час обробки запиту хуком useMutation усі інпути та інтерактивні елементи інтерфейсу блокуються з відображенням процесу завантаження, захищаючи систему від повторних кліків. Якщо сервер повертає помилку автентифікації, у межах body-блоку миттєво з'являється локалізоване повідомлення про збій, а в разі успішного оновлення даних - додаток повністю очищує кеш запитів queryClient, викликає метод logout() для деавторизації

користувача в Zustand-сховищі, скидає поля форми та примусово перенаправляє нас на сторінку входу для авторизації вже з новими безпечними реквізитами.

ЗМІНА ПАРОЛЮ	
ПАРОЛЬ	
ВВЕДІТЬ НОВИЙ ПАРОЛЬ	
ВВЕДІТЬ НОВИЙ ПАРОЛЬ ВДРУГЕ	
ОЧИСТИТИ ВІДПРАВИТИ	

Рис. 3.14 - Сторінка зміни пароля

Спробуємо зайти на сторінку отримання ролі учня, але оскільки відповідна роль у користувача вже присутня, то буде відображено сторінку забороненого доступу (рис. 3.15), котра з'являється при спробі користувача без необхідної ролі увійти на сторінку без відповідних прав.

403

ДОСТУП ЗАБОРОНЕНО

← НАТИСНІТЬ СЮДИ ЩОБ
 ПОВЕРНУТИСЬ НА ГОЛОВНУ
 СТОРІНКУ

Рис. 3.15 - Сторінка забороненого доступу

Відповідно, якщо ролі учня нема і користувач авторизований, то відповідна форма для отримання ролі буде відображуватись (рис. 3.16). Використано як клієнтську, так і серверну валідацію полів.

TITLE	
ІМ'Я ТА ПРИЗВИЩЕ	Імя Прізвище< ✖ Невірний формат імені
ЦІЛЬ НАВЧАННЯ	Імя ПрізвищеІмя Прізвище ✖ Використання HTML-тегів та символів < або > заборонено
ДАТА НАРОДЖЕННЯ	27.06.2026 ✖ Дата народження не може бути в майбутньому
ВІДМІНИТИ ЗМІНИ ЗБЕРЕГТИ ЗМІНИ	

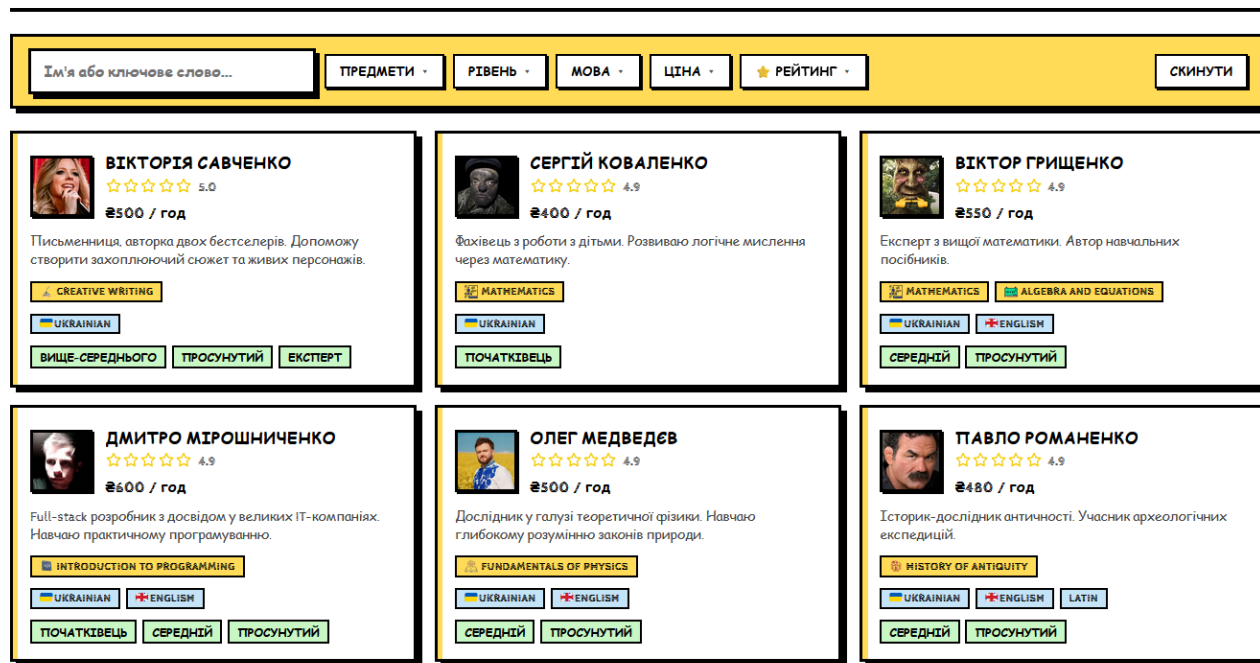
Рис. 3.16 - Сторінка створення акаунта учня з валідацією

При введенні користувачем в рядок посилання шляху, якого нема на сайті буде відображено сторінку помилки 404 з посиланням (рис. 3.17), котре поверне його на головну сторінку.



Рис. 3.17 - Сторінка помилки 404

Для пошуку вчителів у користувача є декілька варіантів, першим із них є сторінка з усіма вчителями на платформі (рис. 3.18). UX сторінки побудований на принципі безперервної взаємодії без перезавантаження вкладки. Користувач вводить текст або обирає параметри у випадваючих вікнах, а система миттєво

ВСІ ВЧИТЕЛІ 71 ЗНАЙДЕНО

Фільтрація та пагінація повністю перенесені на сторону клієнта, що забезпечує нульову затримку при перемиканні екранів. Усі операції порівняння, пошуку за підрядком та сортування за ціною чи рейтингом виконуються над локальним масивом даних, який при кожній зміні конфігурації скидає індекс поточної сторінки на початковий. Фінальний результат маніпуляцій розділяється на порції строго по дев'ять об'єктів за допомогою методу slice, а нижній навігаційний блок динамічно перераховує загальну кількість сторінок, блокує неактивні стрілки переходу на граничних позиціях та підсвічує поточний маркер.

Другим варіантом пошуку вчителя є через сторінку відповідного предмета, для того, щоб на неї перейти користувачу слід спочатку обрати категорію навчальних дисциплін.

Взаємодія зі сторінкою базується на принципі миттєвого зворотного зв'язку та безпечного контекстного керування: доки асинхронний хук завантажує дані, користувач бачить лоадер Spinner, а у випадку рендерить екран помилки NoneCategory. Основна логіка зосереджена на сітці карток, які фізично реагують на курсор миші зміщенням по осях та збільшенням тіні, чітко сигналізуючи про свою клікабельність, тоді як для адміністраторів на картки додається кнопка редагування, ізольована методом `e.stopPropagation()` для запобігання випадкового переходу за посиланням самого предмета. Клієнтська маршрутизація та фільтрація працюють у синергії з адресним рядком, оскільки компонент автоматично зчитує категорію з URL, трансформує пробіли в нижні підкреслення для сумісності з API запитам, а при виборі конкретної дисципліни здійснює зворотну конвертацію тексту в дефіси для роутингу та формування посилань.

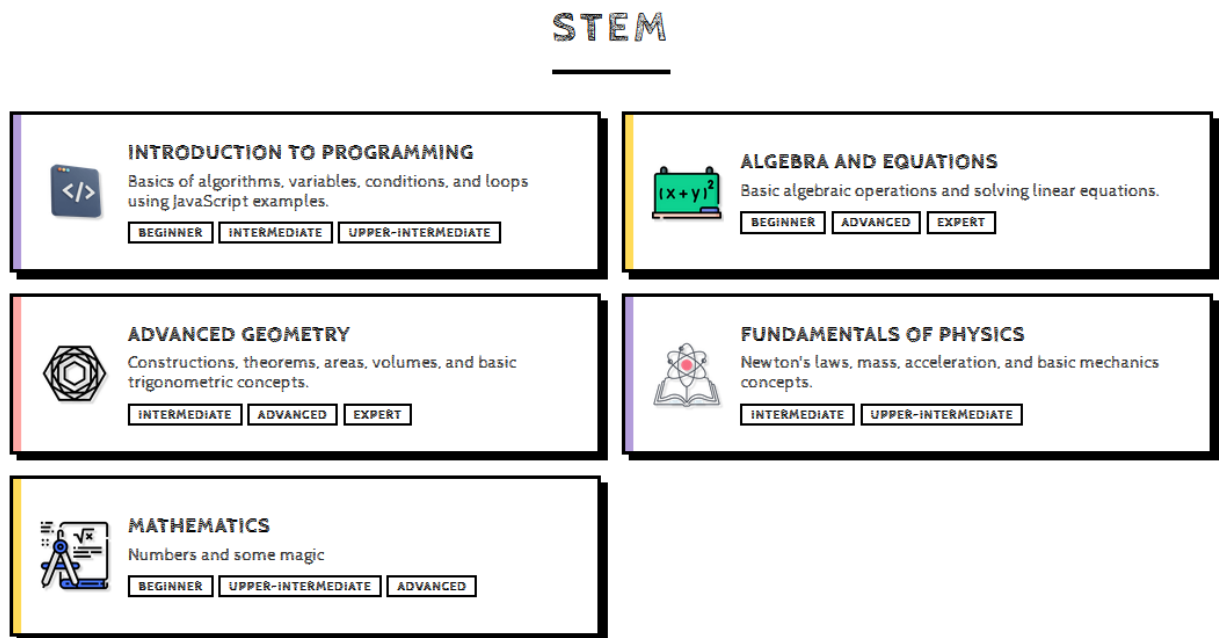


Рис. 3.19 - Сторінка категорії

Коли користувач натискає на відповідний предмет, його перенаправляє на відповідну сторінку (рис. 3.20). Розроблений програмний компонент сторінки предметів реалізує функціонал динамічного формування та відображення переліку викладачів, які спеціалізуються на обраній користувачем дисципліні. Базуючись на параметрах маршрутизації, модуль ініціює асинхронний запит до

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						79

серверної частини для отримання масиву профільних даних фахівців. Архітектура сторінки передбачає комплексну обробку станів життєвого циклу запиту. Візуальне представлення даних організовано у вигляді адаптивного списку інтерактивних карток, кожна з яких агрегує ключові відомості про викладача і слугує навігаційним елементом для переходу до детальної сторінки профілю обраного спеціаліста.

FUNDAMENTALS OF PHYSICS



ОЛЕГ МЕДВЕДЕВ

БІОГРАФІЯ Дослідник у галузі теоретичної фізики. Навчаю глибокому розумінню законів природи.

ДОСВІД 18 років

ОСВІТА Доктор фізико-математичних наук

500 € / год

ЛАРИСА ФЕДОРЕНКО

БІОГРАФІЯ Спеціаліст з експериментальної фізики. Багато практичних демонстрацій.

ДОСВІД 14 років

ОСВІТА Кандидат фізико-математичних наук

450 € / год





РОМАН ШЕВЧУК

БІОГРАФІЯ Інженер-фізик. Навчаю застосуванню фізики в реальному житті.

ДОСВІД 9 років

ОСВІТА Магістр технічної фізики

Рис. 3.20 - Сторінка предмету

Відповідно через обидва шляхи пошуку вчителя учень може потрапити на сторінку спеціаліста (рис. 3.21). Цей програмний компонент функціонує як деталізована візитна картка, що консолідує вичерпні відомості про кваліфікацію, академічний досвід, мовні компетенції, цільові рівні підготовки та фінансові умови співпраці з обраним репетитором. Для забезпечення прозорості якості послуг у структуру інтерфейсу інтегровано підсистему соціального доказу, яка виводить історію текстових рецензій та кількісних рейтингових оцінок від попередніх здобувачів освіти. Ключовою інтерактивною можливістю модуля є

механізм ініціації навчання, реалізований у вигляді модального вікна для подання первинної заявки.



ГАЛИНА САВЧУК

★ 4

РЕЙТИНГ

390 ₴

₴/ГОД

ПРО ВИКЛАДАЧА

БІОГРАФІЯ

Економістка для новачків. Просто про складне.

ОСВІТА

Магістр економіки!

ДОСВІД

5 років

ВИКЛАДАННЯ

МОВИ

English

РІВНІ

Beginner

BASIC

INTERMEDIATE

UPPER_INTERMEDIATE

ADVANCED

ВІДГУКИ УЧНІВ 1

P

РОМАН ФЕЛАТЧУК ★★★★★

навчання імцумцум

MAY 19, 2026

ГОТОВІ ПОЧАТИ?

ЗАЛИШИТИ ВІДГУК

ЗАБРОНЮВАТИ УРОК →

Рис. 3.21 - Сторінка викладача

Даний елемент містить форму комунікації зі строгою багаторівневою клієнтською валідацією (рис. 3.22), яка не лише контролює допустимий обсяг повідомлення, але й виконує санітизацію вхідних даних перед безпечним асинхронним передаванням запитом до сервера.

У результаті успішної ініціації запиту на співпрацю відповідний транзакційний запис реєструється в системі: для здобувача освіти він стає доступним у спеціалізованому модулі моніторингу вихідних заявок, тоді як викладач отримує його у своєму робочому кабінеті.

ПОДАТИ ЗАЯВКУ



ГАЛИНА САВЧУК

★ 4 390 ₪/год

ПОВІДОМЛЕННЯ ДЛЯ ВЧИТЕЛЯ

Я хочу навчатись у вас. Бажані теми для навчання: глобальна економіка, с/г сектор. Бажані години навчання 08:00-17:00 понеділок та вівторок.

СКАСУВАТИ

НАДІСЛАТИ →

Рис. 3.22 - Форма подачі заявки на навчання

Програмний компонент сторінки заявок учня функціонує як комплексна персональна панель для моніторингу та управління життєвим циклом сформованих звернень (рис. 3.23). Інтерфейс агрегує дані у вигляді структурованих карток, які містять профіль репетитора, текст повідомлення, дату створення, а також динамічний статус обробки із кольоровим маркуванням та відображенням аргументованої причини у разі відмови. Окрім пасивного відстеження результатів, модуль надає інструментарій активної взаємодії, дозволяючи користувачу анулювати ще не розглянуті заявки. Для запобігання випадковим деструктивним діям цей механізм реалізовано через проміжне діалогове вікно підтвердження, після затвердження якого ініціюється безпечний асинхронний запит до серверної частини із подальшим реактивним оновленням клієнтського стану інтерфейсу.

МОЇ ЗАЯВКИ

2 ЗАЯВОК



ГАЛИНА САВЧУК

★ 4 390 ₪/год 6/3/2026

ОЧІКУЄ

СКАСУВАТИ

ВАШЕ ПОВІДОМЛЕННЯ:

Я хочу навчатись у вас. Бажані теми для навчання: глобальна економіка, с/г сектор. Бажані години навчання 08:00-17:00 понеділок та вівторок.



ГАЛИНА САВЧУК

★ 4 390 ₪/год 5/27/2026

ВІДХИЛЕНО

ПРИЧИНА ВІДМОВИ:
Not interested

ВАШЕ ПОВІДОМЛЕННЯ:

ifqwoefqwe fwefjn qwoekf

Рис. 3.23 - Приклад надісланих щаявок одним студентом

Для логічного завершення циклу комунікації на платформі розроблено модуль обробки вхідних заявок, який функціонує в межах особистого кабінету викладача (рис. 3.24). Даний інтерфейс виступає в ролі спеціалізованої підсистеми управління взаємовідносинами, надаючи фахівцю інструменти для ефективної класифікації та адміністрування звернень за допомогою категоризації за їхнім поточним станом. Кожна інформаційна картка розкриває розширений контекст про потенційного здобувача освіти, деталізуючи його персональну навчальну мету, вікові параметри та текст первинного запиту.

Процедура прийняття рішень репетитором базується на двох ключових бізнес-транзакціях: безпосередньому ухваленні співпраці або її ініційованому скасуванні. З метою підтримки високих стандартів академічної етики, алгоритм відмови вимагає обов'язкового надання аргументованого зворотного зв'язку через відповідне діалогове вікно. Цей механізм супроводжується посиленням клієнтським контролем валідації даних: форма відхилення програмно перевіряє текст на змістовність та автоматично блокує спроби інтеграції сторонніх веб-посилань, використання заборонених синтаксичних конструкцій чи надлишкових пробілів, гарантуючи отримання студентом виключно конструктивної відповіді для подальшого коригування своїх пошуків.

ВХІДНІ ЗАЯВКИ
1 НОВИХ

ВСІ
ОЧІКУЮТЬ
ПРИЙНЯТІ
ВІДХИЛЕНІ

LALKAZAURUS COMMON
ОЧІКУЄ

МЕТА:
I wanna study something
ДАТА НАРОДЖЕННЯ:
11/24/2003
ПОДАНО:
6/3/2026

ПОВІДОМЛЕННЯ ВІД УЧНЯ:
 Я хочу навчатись у вас. Бажані теми для навчання: глобальна економіка, с/г сектор. Бажані години навчання 08:00-17:00 понеділок та вівторок.

ПРИЙНЯТИ
ВІДХИЛИТИ

МИКИТА СТУДЕНТ
ПРИЙНЯТО

МЕТА:
ВЧИТИСЯ ХОЧУ ДУЖЕ СИЛЬНО
ДАТА НАРОДЖЕННЯ:
3/13/2003
ПОДАНО:
6/1/2026

ПОВІДОМЛЕННЯ ВІД УЧНЯ:
 Доброго дня я хооооочу урок

Рис. 3.24 - Кабінет заявок у вчителя

Після підтвердження заявки вчитель та учень утворюють зв'язок в БД і на відповідних сторінках відображаються один у одного. Для здобувача освіти цей функціонал реалізовано у вигляді спеціалізованого модуля «Мої викладачі», що

акумулює перелік затверджених репетиторів. Інтерфейс сторінки генерує структуровану сітку інформаційних карток, кожна з яких презентує ключові відомості про спеціаліста: фотографію, повне ім'я, коротку біографічну довідку, актуальний рейтинг та вартість послуг (рис. 3.25).

Ключовою інтерактивною особливістю даного компонента є можливість надання зворотного зв'язку про якість освітніх послуг. Кожна картка викладача оснащена тригером ініціації відгуку, який викликає модальне вікно з формою оцінювання. Система передбачає виставлення кількісної оцінки за п'ятибальною шкалою, реалізованою через інтерактивний компонент з рейтинговими зірками, та додавання розгорнутого текстового коментаря. Як і в попередніх модулях платформи, форма відгуку забезпечена суворою клієнтською валідацією, що перевіряє довжину тексту та блокує використання сторонніх посилань чи недопустимих символів, гарантуючи чистоту та безпеку інформації перед відправленням до бази даних і наступним реактивним оновленням рейтингів на платформі.

МОЇ ВЧИТЕЛІ 1 ВЧИТЕЛІВ



Рис. 3.25 - Сторінка вчителів учня

Симетрично до інтерфейсу здобувача освіти, робочий простір викладача містить програмний модуль «Мої учні», призначений для консолідованого моніторингу активного контингенту слухачів. Ця сторінка візуалізує реєстр затверджених до співпраці користувачів у вигляді лаконічних карток, які відображають ініціали, повне ім'я та ключову навчальну мету кожного підопічного. Для забезпечення двостороннього та об'єктивного характеру репутаційної моделі платформи, даний компонент наділяє фахівця інструментарієм зворотного оцінювання (рис. 3.26). За допомогою відповідного тригера викликається модальне вікно рецензування, де репетитор має змогу

охарактеризувати рівень залученості та успішність студента, виставивши бальну оцінку та сформувавши текстовий коментар. Надійність зібраної інформації підтримується уніфікованим механізмом багатокритеріальної валідації, який автономно відфільтровує нерелевантний контент, спам-елементи та заборонені символи ще на етапі введення даних.

МОЇ СТУДЕНТИ

Студентів: 4

Н	НОВІНСЬКИЙ УЧЕНЬСЛАВ Буду дуже старанно вчитись	ВІДГУК
Р	РОМАН ФАЛАПЧУК Вивчити та і забути згодом	ВІДГУК
L	LALKAZARUS COMMON I wanna study something	ВІДГУК
М	МИКИТА СТУДЕНТ ВЧИТИСЯ ХОЧУ ДУЖЕ СИЛЬНО	ВІДГУК

Рис. 3.26 - Сторінка учнів вчителя

Для централізованого управління репутаційним профілем користувача розроблено модуль «Мої відгуки» (рис. 3.27), який консолідує всю історію оцінювання в межах єдиного інтерфейсу. Структурно сторінка поділена на дві логічні вкладки: перша відображає відгуки, створені безпосередньо авторизованим користувачем, тоді як друга агрегує рецензії, отримані ним від інших учасників освітнього процесу. Кожен запис репрезентовано у вигляді деталізованої інформаційної картки, що містить ідентифікатор автора або адресата, кількісний показник рейтингу, текстовий коментар та відповідні рольові маркери, що забезпечує чітке розуміння напрямку комунікації.

МОЇ ВІДГУКИ

НАПИСАНІ 2		ОТРИМАНІ 1	
Н	НОВІНСЬКИЙ УЧЕНЬСЛАВ	РЕДАГУВАТИ	ВИДАЛИТИ
	Студент ★★★★★ Прекрасний контакт		
Р	РОМАН ФАЛАПЧУК	РЕДАГУВАТИ	ВИДАЛИТИ
	Студент ★★★★★ Дуже приємний учень		

Рис. 3.27 - Сторінка відгуків

Ключова функціональна відмінність між вкладками полягає в рівнях доступу: отримані відгуки доступні виключно для ознайомлення, натомість для написаних передбачено повний спектр операцій управління. Механізм редагування імплементовано через виклик модального вікна (рис. 3.28), що містить поточні значення рейтингу та тексту з подальшою їх модифікацією за умови успішного проходження клієнтської валідації. Для операції видалення інтегровано проміжний етап підтвердження, який убезпечує користувача від випадкових втрат даних. Усі транзакції зміни стану супроводжуються асинхронними запитами до серверної частини із наступним реактивним перемальовуванням інтерфейсу та оновленням кількісних індикаторів на навігаційних вкладках.

РЕДАГУВАТИ ВІДГУК

Н

НОВІНСЬКИЙ УЧЕНЬСЛАВ

ОЦІНКА

★★★★★

КОМЕНТАР

Прекрасний контакт

СКАСУВАТИ

ЗБЕРЕГТИ

Рис. 3.28 - Редагування коментаря

Сторінка TeacherLessons забезпечує викладачу робоче місце для управління графіком, де він може перемикатися між переглядом лише найближчих занять та повною історією своїх уроків (рис. 3.29). На кожній картці користувач бачить вичерпні дані про сесію: тему, навчальний предмет, статус оплати, ім'я учня, точний час проведення з обчисленою тривалістю, зафіксовану вартість, домашнє завдання та пряме посилання для швидкого переходу на онлайн-зустріч. Інтерфейс надає повний спектр дій для адміністрування: викладач може створювати нові заняття з нуля за допомогою кнопки додавання, коригувати деталі наявних уроків та безповоротно видаляти записи через вікно підтвердження, яке захищає від випадкових дій і самостійно оновлює екран відразу після видалення. Окремим інструментом є можливість формування звіту про виконання, яка стає доступною виключно для занять, що вже фактично завершилися, дозволяючи викладачу підбивати підсумки уроків, тоді як вбудовані індикатори завантаження та сповіщення про помилки допомагають користувачу завжди бачити актуальний стан системи.

МОЇ УРОКИ

МАЙБУТНІ

ВСІ

+ НОВИЙ УРОК

PRESENT PERFECT

Economics for Beginners

✓ Оплачено

ЗВІТ

РЕДАГУВАТИ

ВИДАЛИТИ

СТУДЕНТ

ПОЧАТОК

ТРИВАЛІСТЬ

ЦІНА

Роман Фалапчук

05/14/2026, 04:30 AM

1г 15хв

888 ₴

Посилання на зустріч

КРУТИЙ УРОК

Economics for Beginners

Не оплачено

ЗВІТ

РЕДАГУВАТИ

ВИДАЛИТИ

СТУДЕНТ

ПОЧАТОК

ТРИВАЛІСТЬ

ЦІНА

Роман Фалапчук

05/17/2026, 05:45 AM

1г 0хв

2131 ₴

Посилання на зустріч

PRESENT PERFECT

Political Systems of the World

Не оплачено

ЗВІТ

РЕДАГУВАТИ

ВИДАЛИТИ

СТУДЕНТ

ПОЧАТОК

ТРИВАЛІСТЬ

ЦІНА

Роман Фалапчук

05/22/2026, 10:30 PM

2г 0хв

50 ₴

Посилання на зустріч

Рис. 3.29 - Сторінка управління уроками

Студент використовує свій кабінет переважно для тайм-менеджменту та розрахунків - відстежуючи зворотний відлік до початку уроку, переглядаючи домашні завдання й сплачуючи борги через шлюзи Stripe чи WayForPay (рис. 3.30), -викладач отримує інструментарій для повного адміністрування графіку. Він може самостійно створювати та редагувати картки занять, видаляти помилкові записи з миттєвим автоматичним оновленням екрана. Після створення або редагування сутностей уроків учень отримує повідомлення на власну електронну скриньку.

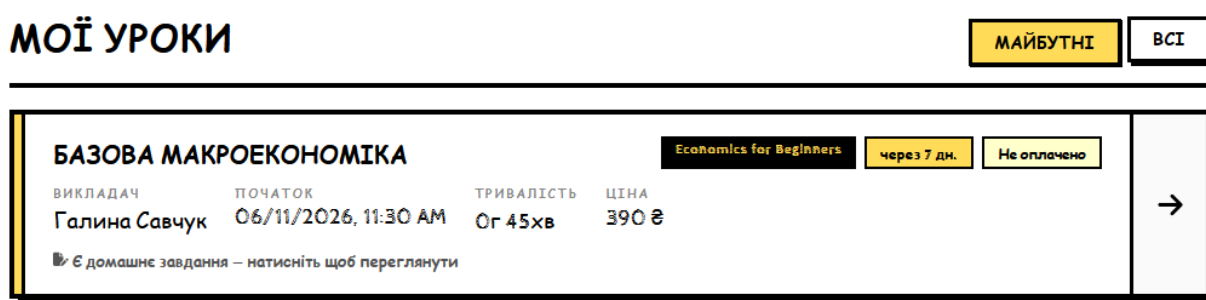


Рис. 3.30 - Сторінка уроків учня

Кожен учень може переглянути власний учнівський акаунт перейшовши туди з сторінки профілю (рис. 3.31).



Рис. 3.31 - Сторінка учнівського профілю

Відповідно з цієї сторінки можна перейти на сторінку редагування поточних учнівських даних, заповнивши відповідну форму (рис. 3.32).

РЕДАГУВАННЯ ПРОФІЛЮ	
ІМ'Я ТА ПРИЗВИЩЕ	Lalkataurus Common
ЦІЛЬ НАВЧАННЯ	I wanna study something
ДАТА НАРОДЖЕННЯ	24.11.2003
ВІДМІНИТИ ЗМІНИ ЗБЕРЕГТИ ЗМІНИ	

Рис. 3.32 - Редагування учнівського профілю

Кожен авторизований користувач має право подати заявку на роботу вчителем, заповнивши відповідну форму (рис. 3.33). Для цього він повинен вказати інформацію, котра надасть розуміння про його професійну діяльність. Також потрібно передати файл власного фото. Після цього заявку буде розглянуто адміністратором.

СТВОРИТИ ПРОФІЛЬ ВЧИТЕЛЯ	
ОСНОВНА ІНФОРМАЦІЯ	
ПОВНЕ ІМ'Я	ЦІНА ЗА ГОДИНУ
ПРО СЕБЕ	
ОСВІТА (ДИПЛОМ)	ДОСВІД РОБОТИ
ФОТО ПРОФІЛЮ	
	ОБРАТИ ФАЙЛ
ПРЕДМЕТИ	
INTRODUCTION TO PROGRAMMING ALGEBRA AND EQUATIONS ADVANCED GEOMETRY HISTORY OF ANTIQUITY	
RENAISSANCE ART ETHICS AND MORAL SYSTEMS LITERARY ANALYSIS SOCIOLOGY OF SOCIETY	
INTRODUCTION TO PSYCHOLOGY ECONOMICS FOR BEGINNERS POLITICAL SYSTEMS OF THE WORLD	
DRAWING BASICS DIGITAL ILLUSTRATION MUSIC CREATION CREATIVE WRITING ENGLISH	
GERMAN UKRAINIAN FOR FOREIGNERS ADVANCED MATH HISTORY OF ROME FUNDAMENTALS OF PHYSICS	
MATHEMATICS	
МОВИ	
UKRAINIAN ENGLISH POLISH GERMAN	
FRENCH SPANISH	
РІВНІ	
BEGINNER BASIC INTERMEDIATE	
UPPER_INTERMEDIATE ADVANCED EXPERT	
ДОСТУПНІ ГОДИНИ	
06:00 07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00	
17:00 18:00 19:00 20:00 21:00 22:00	
ОЧИСТИТИ НАДІСЛАТИ	

Рис. 3.33 - Форма подачі заявки

Для того, щоб викладач почав роботу, представник адміністрації повинен переглянути відповідну заявку (рис. 3.34). Отже він переходить на сторінку з усіма заявками на ролі і відкриває відповідну сторінку. В блоці подання вказана додаткова інформація про нього, для кращого розуміння адміністрацією. У випадку відхилення адміністратор повинен вказати причину у відповідному модальному вікні.

Рис. 3.34 - Подана заявка вчителя

Після підтвердження від адміністрації, користувач отримує роль вчителя і може переглянути власний профіль викладача. Також його профіль стає доступним для перегляду іншими користувачами. На сторінці перегляду акаунта власником видно вказану ним інформацію (рис. 3.35).

Рис. 3.35 - Перегляд профіля вчителя власником

Кожен вчитель може змінити застарілу чи невірно вписану інформацію про себе, заповнивши відповідну форму (рис. 3.36), котра бере за основу його ж дані.

EDIT-TITLE

BASIC INFORMATION

FULL NAME

Петро Петренко

300

BIOGRAPHY

Вульгаризм 20 років, викладач математики, підготовка до НМТ

DEGREE

Кієвський педагогічний університет

EXPERIENCE

20 років

PROFILE PHOTO



CHOOSE FILE

SUBJECTS

INTRODUCTION TO PROGRAMMING

ALGEBRA AND EQUATIONS

ADVANCED GEOMETRY

HISTORY OF ANTIQUITY

HERMENEUTICS

ETHICS AND MORAL SYSTEMS

LITERARY ANALYSIS

SOCIOLOGY OF SOCIETY

INTRODUCTION TO PSYCHOLOGY

ECONOMICS FOR BEGINNERS

POLITICAL SYSTEMS OF THE WORLD

DRAWING BASICS

DIGITAL ILLUSTRATION

MUSIC CREATION

CREATIVE WRITING

ENGLISH

GERMAN

GERMAN FOR FOREIGNERS

ADVANCED PATH

HISTORY OF BOOKS

FUNDAMENTALS OF PHYSICS

INTERMEDIATE

LANGUAGES

GERMAN

ENGLISH

POLISH

RUSSIAN

RUSSIAN

ENGLISH

LEVELS

BEGINNER

BASIC

INTERMEDIATE

UPPER_INTERMEDIATE

ADVANCED

EXPERT

AVAILABLE TIMES

06:00

07:00

08:00

09:00

10:00

11:00

12:00

13:00

14:00

15:00

16:00

17:00

18:00

19:00

20:00

21:00

22:00

RESET

SUBMIT

Рис. 3.36 - Сторінка редагування профілю вчителя

Після проведення уроку, вчитель зобов'язаний заповнити звіт з уроку, разом із скріншотом-підтвердженням відеоконференції з учнем. Також він вказує додаткову інформацію, щоб адміністрація краще розуміла як проходить навчальний процес. Згодом представники школи переглядають звіти на відповідній сторінці (рис. 3.37), фільтруючи їх за потрібними показниками.

ЗВІТИ УРОКІВ

2 / 2 звітів

Всі вчителі ▾
Всі студенти ▾
Всі предмети ▾

Будь-яка відвідуваність ▾

↓ Дата ↓ ▾

КРУТИЙ УРОК					Economics for Beginners	✓ Присутній
ВЧИТЕЛЬ	СТУДЕНТ	ДАТА УРОКУ	ТРИВАЛІСТЬ	ЗВІТ СТВОРЕНО		
Галина Савчук	Роман Жолалчук	05/17/2026, 05:45 AM	1год 0хв	05/22/2026, 07:28 PM		
Content: Основи Present Perfect						

PRESENT PERFECT					Economics for Beginners	✓ Присутній
ВЧИТЕЛЬ	СТУДЕНТ	ДАТА УРОКУ	ТРИВАЛІСТЬ	ЗВІТ СТВОРЕНО		
Галина Савчук	Роман Жолалчук	05/14/2026, 04:30 AM	1год 15хв	05/22/2026, 07:42 PM		
Content: Present Perfect основи						

Рис. 3.37 - Сторінка перегляду звітів

Для постійного підтримування розширення можливих навчальних дисциплін адмін має можливість додавати нові предмети, заповнивши відповідну форму (рис. 3.38).

Рис. 3.38 - Форма додавання предмету

Якщо ж якісь дані предмету застаріли, то адмін може їх оновити, для цього йому потрібно натиснути на кнопку редагування на сторінці категорії предметів і заповнити форму (рис. 3.39).

Рис. 3.39 - Форма оновлення предмету

На платформі використовуються одразу декілька списків змінних, котрі адмін повинен підтримувати в актуальному стані: рівні викладання дисциплін, мови викладання та категорії предметів, ці всі списки він може переглядати та редагувати на відповідній сторінці (рис. 3.40).

Рис. 3.40 - Сторінка змін списків

Для отримання ролей модератора або адміністратора, користувачі потрібно заповнити заявку (рис. 3.41), детально вказавши власну мотивацію для цього. Перегляд заявки відбувається аналогічно з роллю вчителя.

Рис. 3.41 - Заповнення анкети на отримання ролі

Основною діяльністю модератора є перегляд коментарів, котрі залишають користувачі на ресурсі і блокування або розблокування користувачів. Для цього сформовано відповідну сторінку (рис. 3.42).

REVIEWS MODERATION

FROM STUDENTS

FROM TEACHERS

2 / 2 reviews

Search by teacher, student, comment...

Any rating

Any status

All reviews

TEACHER

★★★★☆

5/19/2026

Дуже приємний учень

TEACHER (AUTHOR)

oleksandr_k

Active

ABOUT STUDENT

roman_f

Active

BAN AUTHOR

BAN SUBJECT

TEACHER

★★★★★

5/28/2026

Прекрасний контакт

TEACHER (AUTHOR)

oleksandr_k

Active

ABOUT STUDENT

Lalkazaurus

Active

BAN AUTHOR

BAN SUBJECT

Рис. 3.42 - Заповнення анкети на отримання ролі

Дії ж адміністраторів та модераторів регулює власник ресурсу, в його можливостях є перегляд діяльності адміністрації загалом і перегляд їх рольового доступу, при невірному виконанні обов’язків адміністраторські ролі забираються (рис. 3.43).

АУДИТ ЛОГІВ

ID КОРИСТУВАЧА

напр. 42

РОЛЬ АКТОРА

Всі ролі

ДІЯ

напр. REMOVE_ROLE

СУТНІСТЬ

Всі сутності

ДАТА З

дд.мм.рррр

ДАТА ПО

дд.мм.рррр

СКИНУТИ

23 записів

ID	ДАТА	АКТОР	РОЛЬ	ДІЯ	СУТНІСТЬ	ID СУТНОСТІ	IP	ДЕТАЛІ	ЗАБРАТИ
23	6/4/2026, 12:49:19 AM	Lalkazaurus	TEACHER	UPDATE_TEACHER_PROFILE	TeacherProfile	94	-	▼	-TEACHER
22	6/4/2026, 12:49:08 AM	Lalkazaurus	TEACHER	UPDATE_TEACHER_PROFILE	TeacherProfile	94	-	▼	-TEACHER
21	6/4/2026, 12:48:18 AM	Lalkazaurus	TEACHER	CREATE_TEACHER_PROFILE	TeacherProfile	94	-	▼	-TEACHER
20	6/3/2026, 1:49:51 PM	b0tt0w	TEACHER	UPDATE_TEACHER_PROFILE	TeacherProfile	93	-	▼	-TEACHER
19	6/2/2026, 1:17:03 AM	b0tt0w	TEACHER	UPDATE_TEACHER_PROFILE	TeacherProfile	93	-	▼	-TEACHER
18	6/2/2026, 12:59:49 AM	Егор Зорембицкий	TEACHER	UPDATE_TEACHER_PROFILE	TeacherProfile	91	-	▼	-TEACHER -MODERATOR -ADMIN
17	6/2/2026, 12:58:06 AM	Егор Зорембицкий	TEACHER	UPDATE_TEACHER_PROFILE	TeacherProfile	91	-	▼	-TEACHER -MODERATOR -ADMIN
16	6/2/2026, 12:45:33 AM	Егор Зорембицкий	TEACHER	UPDATE_TEACHER_PROFILE	TeacherProfile	91	-	▼	-TEACHER -MODERATOR -ADMIN

Рис. 3.43 - Сторінка логів

Також для ведення фінансового аудиту власнику доступна сторінка з усіма проведеними платежами на ресурсі (рис. 3.44).

ОПЛАТИ

11 з 11 записів

ВСЬОГО ОПЛАТ
11

УСПІШНИХ
3

ОЧІКУЮТЬ
3

ПОМИЛКИ
5

ПОВЕРНЕНО
0

ЗІБРАНО (UAH)
978,00

Пошук по користувачу, уроку, ID транзакції...

BCI

УСПІШНО

ОЧІКУЄ

ПОМИЛКА

ПОВЕРНЕНО

BCI

STRIPE

WAYFORPAY

↑ Нові спочатку

ID	КОРИСТУВАЧ	УРОК	СУМА	ПРОВАЙДЕР	СТАТУС	ДАТА
#48	Lalkazaurus indiraptor63983@gmail.com	Базова макроекономіка	390,00 UAH	Stripe	Очікує	04.06.2026, 00:18
#40	Mykyta chornous.vlad2005@gmail.com	Вчим Нікіту думать	490,00 UAH	WayForPay	Очікує	01.06.2026, 14:32
#39	roman_f roman.f@example.com	Презент сімпл	55,00 UAH	Stripe	Успішно	25.05.2026, 17:25

Рис. 3.44 - Сторінка оплат

В кінці виконання власної роботи користувач може вийти з акаунту натиснувши відповідну кнопку “Вийти” у випадяючому меню профілю.

3.3. Тестування роботи програмного забезпечення

Тестування розробленого програмного забезпечення є критично важливим етапом життєвого циклу розробки, який дозволяє переконатися у повній відповідності реалізованого функціоналу заявленим технічним, бізнес- та користувацьким вимогам. З огляду на архітектурні особливості створеного вебзастосунку, для комплексної перевірки працездатності його клієнтської частини було обрано методологію ручного функціонального тестування методом «чорної скриньки». Такий підхід дозволяє найточніше змоделювати реальну поведінку кінцевих користувачів - студентів та викладачів.

Першим важливим етапом перевірки є тестування форм введення даних та їх клієнтської валідації на прикладі сторінки створення профілю. Під час ручного тестування навмисно вводяться некоректні значення, такі як спроби використання HTML-тегів у текстових полях, вставлення заборонених веб-посилань, надто короткі описи або вибір майбутніх дат народження. Очікуваним

результатом є миттєве блокування відправки форми системою з одночасним виведенням відповідних червоних локалізованих повідомлень про помилку безпосередньо під кожним полем, що надійно запобігає потраплянню невалідних даних на сервер. Наступний скріншот демонструє реакцію інтерфейсу на спробу збереження анкети із заздалегідь некоректно заповненими даними (рис. 3.45).

The screenshot shows a web form titled "ADD TEACHER PROFILE". It contains several input fields with validation errors indicated by red text below them:

- BASIC INFORMATION**
 - FULL NAME:** The input field contains the HTML code `<h1>Єгор Зарембицький</h1>`. Below it is the error message: "X Invalid name format".
 - PRICE PER HOUR:** The input field contains `-30`. Below it is the error message: "X Minimum price is 1".
- BIOGRAPHY:** The input field contains `ієррріє`. Below it is the error message: "X Biography must be at least 20 characters".
- DEGREE:** The input field contains `years: 2` and `university: "КПІ"`. Below it is the error message: "X invalid-chars".
- EXPERIENCE:** The input field contains a long URL starting with `https://docs.google.com/spreadsheets/d/1j8olx1e-HYsXY9_TAY17UaONmPloUllDsC4Tuj2LwKg/edit?gid=1585750015#gid=1585750015`. Below it is the error message: "X no-urls".
- PROFILE PHOTO:** There is a placeholder image and a "CHOOSE FILE" button.

Рис. 3.45 - Валідація форми

Далі перевіряється коректність відображення станів інтерфейсу під час роботи з бізнес-логікою застосунку, зокрема у модулі розкладу занять та під час виконання платіжних операцій. Тестувальник імітує процес взаємодії з картками уроків, звертаючи увагу на блокування елементів керування під час асинхронних запитів: при натисканні на кнопку оплати система повинна миттєво змінити стан на завантаження, активувати відповідний спінер та зробити кнопку неактивною, щоб повністю унеможливити подвійне списання коштів. Водночас перевіряється поведінка модальних вікон і динамічних бейджів, щоб переконатися у відсутності візуальних артефактів. Наведений нижче рисунок ілюструє цей захищений перехідний стан інтерфейсу в момент ініціалізації транзакції (рис. 3.46).

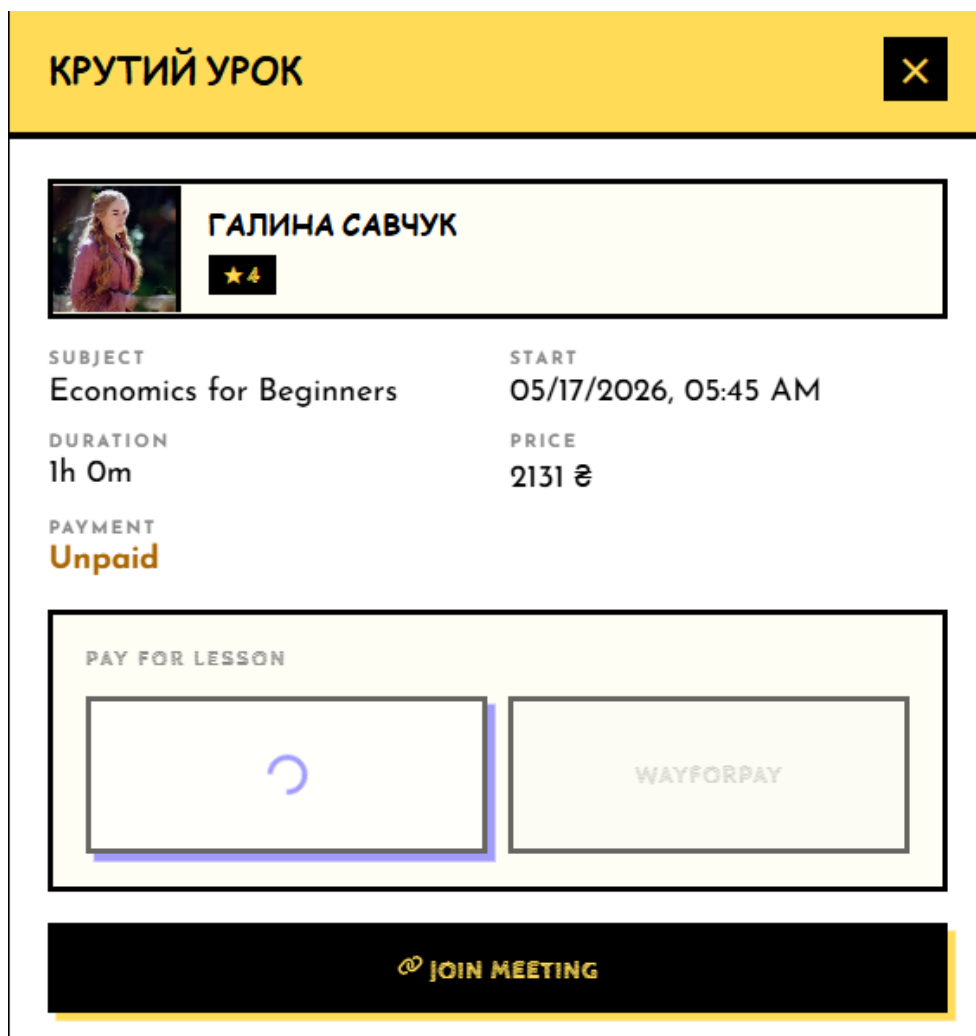


Рис. 3.46 - Блокування оплати в процесі завантаження

Окремої уваги заслуговує перевірка реакції клієнтського застосунку на спроби введення дубльованих даних, що порушують обмеження унікальності на рівні системи. Наприклад, під час тестування свідомо ініціюється реєстрація з електронною адресою, яка вже прив'язана до іншого акаунта. У таких випадках очікується, що фронтенд коректно перехопить негативну відповідь сервера і замість незрозумілого системного збою чи зависання екрана відобразить чітке сповіщення. Це повідомлення має проінформувати користувача про наявність конфлікту записів та запобігти подальшим невалідним діям. Цей тест підтверджує стійкість інтерфейсу до логічних помилок взаємодії з бекендом, а вигляд відповідного попередження про виявлений дублікат показано на наступному скріншоті (рис. 3.47).

REGISTER

× Failed to register

ENTER YOUR E-MAIL

indiraptor63983@gmail.com

ENTER YOUR PASSWORD

.....

ENTER YOUR PHONE NUMBER

+380966747611

ENTER YOUR PHONE USERNAME

noi

RESET SUBMIT

АБО

SIGN UP WITH GOOGLE

DO YOU HAVE AN ACCOUNT? [LOG IN](#)

× USER ALREADY EXISTS

Рис. 3.47 - Спроба реєстрації із вже використовуваною електронною скринькою

Ще одним важливим сценарієм є тестування бізнес-логіки системи щодо специфічних обмежень дій користувача, зокрема перевірка заборони на залишення повторного відгуку. Відповідно до правил платформи, один студент має право написати лише одну рецензію на конкретного викладача. Під час тестування перевіряється ситуація, коли користувач, який вже поділився своїми враженнями, намагається надіслати ще один відгук тому ж репетитору. Коректна поведінка фронтенду у цьому випадку полягає у превентивному приховуванні форми зворотного зв'язку на сторінці профілю викладача або ж у коректному перехопленні серверної помилки з виведенням інформативного спливаючого повідомлення про те, що відгук для цього спеціаліста вже був успішно доданий раніше. Такий підхід не лише надійно захищає рейтингову систему від штучних накруток або емоційного спаму, але й забезпечує прозорий користувацький досвід без незрозумілих системних збоїв. Відповідний стан інтерфейсу з блокуванням можливості повторного оцінювання наведено на рисунку нижче (рис. 3.48).



Рис. 3.48 - Спроба написати другий коментар про вчителя

Заключним етапом тестування клієнтської частини є перевірка системи маршрутизації та механізмів розмежування прав доступу, що критично важливо для безпеки платформи. У цьому контексті перевіряється поведінка застосунку при спробах неавторизованого доступу до захищених сторінок або, наприклад, при спробі користувача з роллю «студент» перейти за прямим посиланням до панелі керування «викладача». Система повинна коректно перехопити таку дію та відмалювати спеціалізовану сторінку заборони доступу (рис. 3.49).

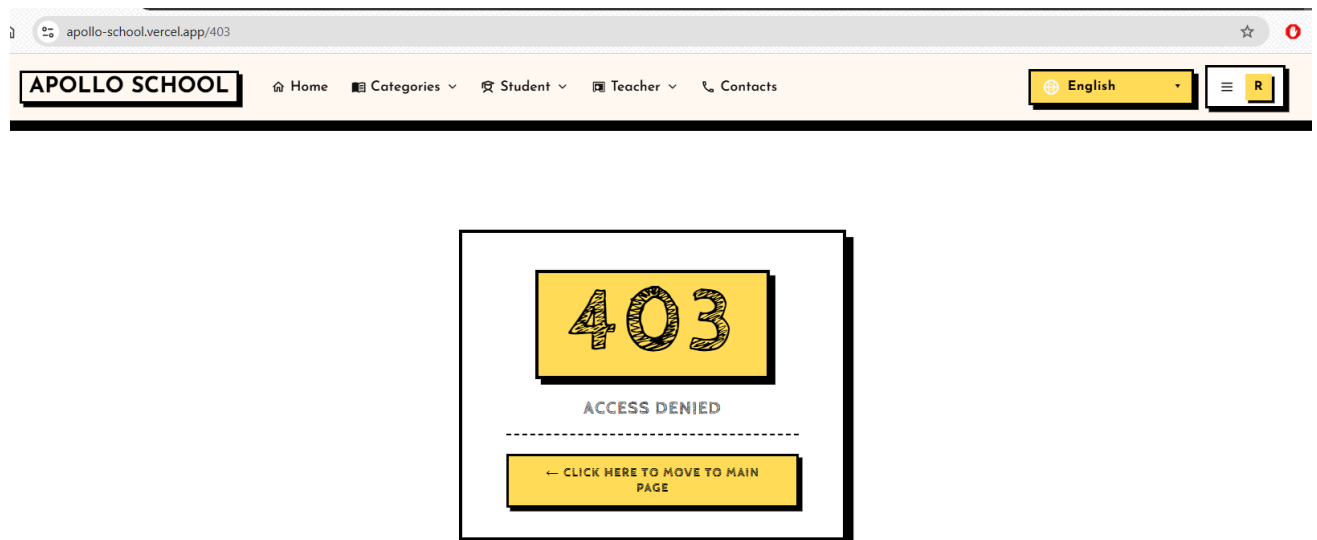


Рис. 3.49 - Заборона для учня перегляду адміністративної панелі вчителя

Крім того, здійснюється перевірка обробки невідомих маршрутів: введення випадкових URL-адрес має гарантовано приводити користувача на сторінку «Не знайдено» зі зрозумілою навігацією для повернення на головну сторінку, а не викликати блокування всієї сторінки. Відповідні стани обробки помилок маршрутизації представлені на наступному зображенні (рис. 3.50).

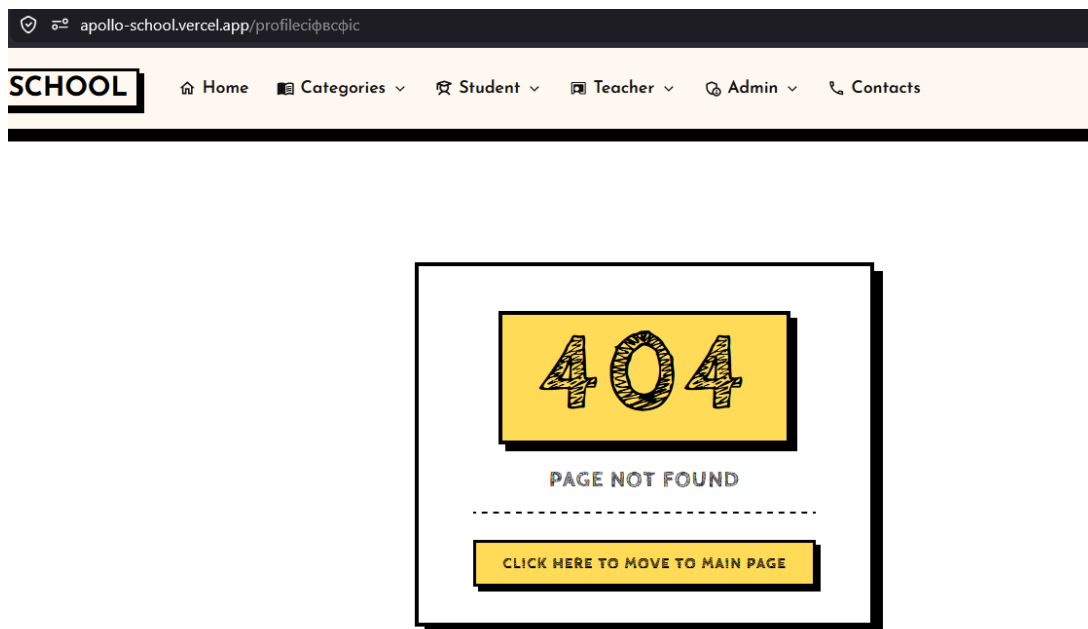


Рис. 3.50 - Обробка неіснуючих маршрутів

Висновки до третього розділу

У третьому розділі було детально розглянуто процес програмної реалізації та налаштування клієнтської частини вебплатформи для організації онлайн-навчання та пошуку репетиторів. Розробка інтерфейсу здійснювалася з акцентом на забезпечення високої реактивності та зручності взаємодії, де кожен функціональний модуль інтегрований у загальну екосистему через надійні механізми синхронізації з бекендом. Такий підхід гарантує не лише стабільність роботи платформи під час виконання асинхронних операцій, а й дозволяє ефективно керувати станами застосунку, що є критично важливим для забезпечення безперебійного освітнього процесу.

У рамках розділу було детально проаналізовано ключовий функціонал застосунку, чітко розділений на спеціалізовані кабінети для різних користувачьких ролей. Для студентів реалізовано зручні інструменти тайм-менеджменту зі зворотним відліком до занять, інтеграцію безпечних платіжних шлюзів Stripe та WayForPay для швидкого розрахунку, а також прямий доступ до онлайн-зустрічей. Зі свого боку, викладачі отримали повноцінний інструментарій для гнучкого адміністрування розкладу та продуману систему формування звітності, яка блокує передчасне заповнення документів до фактичного завершення уроку, чим забезпечує абсолютну прозорість навчання.

Окрему увагу приділено процесам первинного формування акаунтів та строгому розмежуванню прав доступу, що формують фундамент безпеки платформи. Реалізований механізм створення профілю супроводжується безшовним фоновим оновленням авторизаційних токенів, миттєво надаючи користувачу відповідні його ролі права. Надійна система маршрутизації гарантує, що специфічні панелі керування надійно захищені від несанкціонованого доступу, а будь-які спроби переходу за недопустимими посиланнями чи порушення рольової моделі коректно перехоплюються сторінками заборони доступу.

Крім того, було проведено комплексне тестування основних компонентів інтерфейсу методом ручної перевірки. Воно підтвердило коректну роботу багаторівневої клієнтської валідації, ефективного блокування інтерфейсу під час фінансових транзакцій для запобігання дублюванню запитів, а також коректну обробку конфліктів у базі даних. Успішні результати цих перевірок дозволяють з упевненістю стверджувати, що розроблена система повністю готова до експлуатації в реальних умовах і здатна забезпечити надійний користувачький досвід для всіх учасників освітнього процесу.

ВИСНОВКИ

Підсумком проведеного дослідження та практичної роботи стало створення повноцінного клієнтського середовища для освітнього вебсервісу, орієнтованого на взаємодію викладачів та учнів. Вивчивши сильні й слабкі сторони наявних на ринку рішень, вдалося сформувати оптимальну архітектуру та обрати відповідні інструменти. Ядром фронтенд-частини виступила зв'язка React та TanStack Query, що вирішила одразу кілька інженерних завдань: оптимізацію мережових запитів, локальне збереження стану та миттєву візуалізацію змін без перезавантаження браузера. Завдяки цьому будь-які оновлення графіку занять або транзакційні статуси відображаються в реальному часі.

Серверна інфраструктура платформи побудована на базі фреймворку NestJS, що забезпечило чітку модульну архітектуру та високу масштабованість бекенду. Для надійного збереження та управління складною ієрархією інформації - від профілів користувачів до статусів фінансових транзакцій - було обрано реляційну систему управління базами даних PostgreSQL. Взаємодія між серверною бізнес-логікою та базою даних реалізована за допомогою об'єктно-реляційного відображення TypeORM. Використання цієї технології гарантує сувору типізацію, безпечне виконання паралельних транзакцій та ефективну оптимізацію складних запитів, створюючи стійкий до навантажень фундамент, який безперебійно обслуговує всі потреби клієнтського застосунку.

Значну частину уваги було приділено проектуванню специфічних користувацьких інтерфейсів, які відповідають унікальним потребам кожної з ролей. Зокрема, кабінет здобувача освіти оснащений засобами контролю часу, включно з візуальним таймером до старту найближчої лекції. Натомість робочий простір репетитора спроектований як багатопрофільний адміністративний хаб. У ньому реалізовано гнучкий календар занять і строгу систему управління документацією, логіка якої програмно унеможливорює створення підсумкових звітів до моменту фактичного закінчення уроку, гарантуючи тим самим абсолютну прозорість навчання.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						102

Безпека та зручність використання платформи реалізовані на рівні введення початкових даних. Розроблений алгоритм реєстрації та запам'ятовування даних передбачає тіньове оновлення ключів авторизації, завдяки чому система безшовно переводить нового клієнта у статус повноправного користувача з миттєвим перенаправленням у його особисту зону. Фінансовий аспект вирішено шляхом підключення платіжних провайдерів Stripe та WayForPay безпосередньо у спливаючі вікна застосунку. Паралельно з цим налаштовано жорсткий контроль навігації: клієнтські алгоритми превентивно відхиляють спроби переходу до чужих або неіснуючих панелей, своєчасно генеруючи коректні сторінки відмов через помилки доступу чи відсутності контенту замість аварійного зависання застосунку.

Фінальним етапом стала серія мануальних випробувань інтерфейсу за методологією «чорної скриньки». Практичні перевірки довели надійність вбудованих валідаторів і стабільність захисних механізмів, які тимчасово заморожують елементи керування під час обробки фінансових чи мережевих операцій, виключаючи ризик подвійних дій. Також підтверджено правильність реакції системи на порушення бізнес-логіки бази даних, як-от спроби опублікувати кілька оцінок одному спеціалісту. Створений програмний продукт успішно пройшов усі стадії життєвого циклу розробки, цілком відповідає сучасним стандартам і є готовим до розгортання в умовах реального ринку цифрових освітніх послуг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зарембицький Є.Ю., Чижмотря О.В. Специфіка управління глобальним станом клієнтських інтерфейсів освітніх вебплатформ на базі бібліотеки React. Матеріали XVI Міжнародної наукової конференції студентів та молодих вчених «Сучасні інформаційні технології – 2026», м. Одеса, 14-15 травня 2026 р. – Одеса: Національний університет «Одеська політехніка», 2026.
2. Зарембицький Є.Ю., Чижмотря О.В. Використання бібліотеки TanStack Query та мови TypeScript для взаємодії API у вебінтерфейсах онлайн-шкіл. Тези Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених «Тенденції розвитку технологій в автоматизації, електроенергетиці, приладобудуванні та робототехніці» (присвяченої Дню науки), м. Житомир, 13-15 травня 2026 р. – Житомир: Житомирська політехніка, 2026.
3. Зарембицький Є.Ю., Чижмотря О.В. Проєктування архітектури високонавантаженої системи вебплатформи онлайн-школи на базі фреймворку NestJS. Тези Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених «Тенденції розвитку технологій в автоматизації, електроенергетиці, приладобудуванні та робототехніці» (присвяченої Дню науки), м. Житомир, 13-15 травня 2026 р. – Житомир: Житомирська політехніка, 2026.
4. Axios Team. Axios Introduction [Електронний ресурс]. URL: <https://axios-http.com/docs/intro> (дата звернення: 05.06.2026)
5. Firebase Team. Cloud Firestore Documentation [Електронний ресурс]. URL: <https://firebase.google.com/docs/firestore> (дата звернення: 05.06.2026)
6. Firebase Team. Firebase Documentation [Електронний ресурс]. URL: <https://firebase.google.com/docs> (дата звернення: 05.06.2026).
7. JWT Team. JSON Web Token Introduction [Електронний ресурс]. URL: <https://jwt.io/introduction> (дата звернення: 05.06.2026).

8. Mozilla Developer Network. MDN Web Docs - Cross-Origin Resource Sharing (CORS) [Электронный ресурс]. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS> (дата звернения: 05.06.2026).
9. Mozilla Developer Network. MDN Web Docs - CSS: Cascading Style Sheets [Электронный ресурс]. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (дата звернения: 05.06.2026).
10. NestJS Team. NestJS Documentation [Электронный ресурс]. URL: <https://docs.nestjs.com/> (дата звернения: 05.06.2026).
11. npm Team. npm Documentation [Электронный ресурс]. URL: <https://docs.npmjs.com/> (дата звернения: 05.06.2026).
12. PostgreSQL Global Development Group. PostgreSQL Documentation [Электронный ресурс]. URL: <https://www.postgresql.org/> (дата звернения: 05.06.2026).
13. React Team. React Documentation [Электронный ресурс]. URL: <https://react.dev/> (дата звернения: 05.06.2026).
14. React Leaflet Team. React Leaflet Introduction [Электронный ресурс]. URL: <https://react-leaflet.js.org/docs/start-introduction/> (дата звернения: 05.06.2026).
15. TanStack Team. TanStack Query Overview [Электронный ресурс]. URL: <https://tanstack.com/query/latest/docs/framework/react/overview> (дата звернения: 05.06.2026).
16. TypeORM Team. TypeORM Documentation [Электронный ресурс]. URL: <https://typeorm.io/> (дата звернения: 05.06.2026).
17. TypeScript Team. TypeScript Documentation [Электронный ресурс]. URL: <https://www.typescriptlang.org/docs/> (дата звернения: 05.06.2026).
18. Zustand Team. Zustand Introduction [Электронный ресурс]. URL: <https://zustand.docs.pmnd.rs/learn/getting-started/introduction> (дата звернения: 05.06.2026).
19. Amazon Web Services. Serverless Computing [Электронный ресурс]. URL: <https://aws.amazon.com/serverless/> (дата звернения: 05.06.2026).

20. IBM. What is PaaS (Platform-as-a-Service)? [Электронный ресурс]. URL: <https://www.ibm.com/topics/paas> (дата звернения: 05.06.2026).
21. Mozilla Developer Network. MDN Web Docs - 401 Unauthorized [Электронный ресурс]. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/401> (дата звернения: 05.06.2026).
22. Stripe Team. Stripe API Reference [Электронный ресурс]. URL: <https://stripe.com/docs/api> (дата звернения: 05.06.2026).
23. Express Team. Express.js Documentation [Электронный ресурс]. URL: <https://expressjs.com/> (дата звернения: 05.06.2026).
24. Express Team. Multer Middleware [Электронный ресурс]. URL: <https://expressjs.com/en/resources/middleware/multer/> (дата звернения: 05.06.2026).
25. Render Team. Render Documentation [Электронный ресурс]. URL: <https://render.com/docs> (дата звернения: 05.06.2026).
26. Vercel Team. Vercel Documentation [Электронный ресурс]. URL: <https://vercel.com/docs> (дата звернения: 05.06.2026).
27. Neon Team. Neon Serverless Postgres Documentation [Электронный ресурс]. URL: <https://neon.tech/docs> (дата звернения: 05.06.2026).
28. Microsoft Learn. RESTful API design [Электронный ресурс]. URL: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design> (дата звернения: 05.06.2026).
29. React Router Team. React Router Official Documentation [Электронный ресурс]. URL: <https://reactrouter.com/> (дата звернения: 05.06.2026).
30. Mozilla Developer Network. MDN Web Docs - TLS (Transport Layer Security) [Электронный ресурс]. URL: <https://developer.mozilla.org/en-US/docs/Glossary/TLS> (дата звернения: 05.06.2026).

ДОДАТКИ

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						107

Лістинг обробки JWT-токенів авторизації користувача

```

import { BadRequestException, Injectable, UnauthorizedException } from
"@nestjs/common";
import { ConfigService } from "@nestjs/config";
import { PassportStrategy } from "@nestjs/passport";
import { ExtractJwt, Strategy } from "passport-jwt";
import { TokenService } from "src/token/token.service";
import { User } from "src/users/user/entities/user.entity";
import { Status } from "src/users/user/types/status";
import { UserService } from "src/users/user/user.service";

@Injectable()
export class JwtStrategy extends PassportStrategy(Strategy) {
  constructor(
    private configService: ConfigService,
    private userService: UserService,
    private tokenService: TokenService
  ) {
    const secret = configService.get<string>('JWT_SECRET')!;
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      ignoreExpiration: false,
      secretOrKey: secret,
      passReqToCallback: true
    });
  }

  async validate(req: Request, payload: User) {
    const authHeader = req.headers['authorization'];
    const token = authHeader?.replace('Bearer ', '').trim();

    if (!token) {
      throw new UnauthorizedException("No token provided");
    }

    const tokenData = await this.tokenService.findToken(payload.id);

    if (!tokenData) {
      throw new UnauthorizedException("Token not found");
    }

    const isValid = await this.tokenService.compareAccessToken(token,
tokenData.accessTokenHash);
    if (!isValid) {
      throw new UnauthorizedException("Invalid or expired token");
    }

    const user = await this.userService.findByEmail(payload.email);
    if (!user) {
      throw new BadRequestException("User doesn't exist");
    }
  }
}

```

```

    }

    if (user.status === Status.BANNED) {
        throw new UnauthorizedException('User is banned');
    }

    const { password, ...userInfo } = user;
    return userInfo;
}
}

```

Лістинг перехоплювачів axios

```

import axios from "axios";
import type { InternalAxiosRequestConfig } from "axios";
import { getErrorMessage } from "../getErrorMessage";
import { toast } from "react-hot-toast";
import { useAuthStore } from "../store/useAuthStore";

interface CustomAxiosRequestConfig extends InternalAxiosRequestConfig {
    _retry?: boolean;
}

const api = axios.create({
    baseURL: import.meta.env.VITE_API_URL,
});

interface QueueItem {
    resolve: (value: string | null) => void;
    reject: (reason: unknown) => void;
}

let isRefreshing = false;
let failedQueue: QueueItem[] = [];

const processQueue = (error: unknown, token: string | null = null) => {
    failedQueue.forEach((prom) => {
        if (error) {
            prom.reject(error);
        } else {
            prom.resolve(token);
        }
    });
};

failedQueue = [];

api.interceptors.request.use((config: CustomAxiosRequestConfig) => {
    const token = localStorage.getItem("accessToken");
    if (token) {
        config.headers["Authorization"] = `Bearer ${token}`;
    }
    return config;
});

```

```

api.interceptors.response.use(
  (response) => response,
  (error) => {
    const originalRequest = error.config as CustomAxiosRequestConfig;

    if (!originalRequest) {
      return Promise.reject(error);
    }

    if (error.response?.status === 401) {

      if (originalRequest._retry) {
        useAuthStore.getState().logout();
        return Promise.reject(error);
      }

      if (isRefreshing) {
        return new Promise<string | null>((resolve, reject) => {
          failedQueue.push({ resolve, reject });
        })
          .then((token) => {
            originalRequest.headers["Authorization"] = `Bearer ${token}`;
            return api(originalRequest);
          })
          .catch((err) => Promise.reject(err));
      }

      originalRequest._retry = true;
      isRefreshing = true;

      return new Promise((resolve, reject) => {
        const refreshToken = localStorage.getItem("refreshToken");

        if (!refreshToken) {
          useAuthStore.getState().logout();
          processQueue(new Error("No refresh token"), null);
          isRefreshing = false;
          return reject(new Error("No refresh token available"));
        }

        axios.post(
          `${import.meta.env.VITE_API_URL}/auth/refresh`,
          {},
          {
            headers: {
              Authorization: `Bearer ${refreshToken}`,
            }
          }
        )
          .then((response) => {
            const { accessToken, refreshToken: newRefreshToken, user } = response.data;

```

```

        useAuthStore.getState().setLogin(user, {
            accessToken: accessToken,
            refreshToken: newRefreshToken
        });

        api.defaults.headers.common["Authorization"] = `Bearer ${accessToken}`;
        originalRequest.headers["Authorization"] = `Bearer ${accessToken}`;

        processQueue(null, accessToken);
        resolve(api(originalRequest));
    })
    .catch((refreshError) => {
        processQueue(refreshError, null);
        useAuthStore.getState().logout();
        reject(refreshError);
    })
    .finally(() => {
        isRefreshing = false;
    });
});
}

if (error.response?.status !== 401) {
    const message = getErrorMessage(error);
    toast.error(message, {
        style: {
            border: '3px solid black',
            background: '#ffeb3b',
            color: '#000',
            padding: '12px 16px',
            fontWeight: 'bold',
            textTransform: 'uppercase',
            boxShadow: '4px 4px 0px #000',
            borderRadius: '0.5rem',
            transform: 'rotate(-1deg)',
        },
        iconTheme: {
            primary: '#d32f2f',
            secondary: '#fff',
        },
    });
}

return Promise.reject(error);
}
);

export default api;

```

Лістинг перехоплення ролей користувача

```

import { CanActivate, ExecutionContext, Injectable } from "@nestjs/common";
import { Reflector } from "@nestjs/core";
import { ROLES_KEY } from "src/common/decorators/role.decorator";
import { Roles } from "src/users/user/types/roles";

@Injectable()
export class RolesGuard implements CanActivate {
  constructor(private reflector: Reflector) {}

  canActivate(context: ExecutionContext): boolean {
    const requiredRoles = this.reflector.getAllAndOverride<Roles[]>(ROLES_KEY, [
      context.getHandler(),
      context.getClass(),
    ]);
    if (!requiredRoles) {
      return true;
    }
    const { user } = context.switchToHttp().getRequest();
    return requiredRoles.some((role) => user.roles?.includes(role));
  }
}

import { SetMetadata } from "@nestjs/common"
import { Roles } from "src/users/user/types/roles"

export const ROLES_KEY = 'roles'
export const Role = (...roles: Roles[]) => SetMetadata(ROLES_KEY, roles)

```

Лістинг збереження файлів на серверній частині

```

@Post('create')
@UseGuards(JwtAuthGuard, RolesGuard)
@Role(Roles.USER)
@UseInterceptors(FileInterceptor('file', { storage: memoryStorage() }))
async create(
  @Body() teacherDto: TeacheProfileInfoDto,
  @UploadedFile() file: Express.Multer.File,
  @Req() req: Request,
) {
  const isTaken = await this.teacherProfileService.isFullNameTaken(teacherDto.fullName);
  if (isTaken) throw new BadRequestException('Вчитель з таким іменем вже існує');

  const user = req.user as ValidatedPayloadDto;
  const imageUrl = file ? await this.firebaseService.uploadImage(file) : undefined;

  const subjectIds: number[] = teacherDto.subjectIds
    ? (typeof teacherDto.subjectIds === 'string' ? JSON.parse(teacherDto.subjectIds) :
teacherDto.subjectIds)
    : [];
  const availableTimes: string[] = teacherDto.availableTimes
    ? (typeof teacherDto.availableTimes === 'string' ?
JSON.parse(teacherDto.availableTimes) : teacherDto.availableTimes)
    : [];
  const levels: string[] = teacherDto.levels
    ? (typeof teacherDto.levels === 'string' ? JSON.parse(teacherDto.levels) :
teacherDto.levels)
    : [];
  const languages: string[] = teacherDto.languages
    ? (typeof teacherDto.languages === 'string' ? JSON.parse(teacherDto.languages) :
teacherDto.languages)
    : [];

  return await this.roleApplicationService.create(

```

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						113

```
{
  requestedRole: Roles.TEACHER,
  status: 'pending',
  comment: "",
  pendingData: { fullName: teacherDto.fullName, bio: teacherDto.bio, degree:
teacherDto.degree, experience: teacherDto.experience, pricePerHour: teacherDto.pricePerHour,
levels, subjectIds, availableTimes, imageUrl, languages },
},
  user.id,
);
}
```