

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

## **ПОЯСНЮВАЛЬНА ЗАПИСКА**

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

на тему:

«Вебплатформа для онлайн-навчання»

Виконав студент групи ВТ-22-1

ГОЛОВНЯ Євген Миколайович

Керівник роботи:

ЛОБАНЧИКОВА Надія Миколаївна

Рецензент:

ЛОКТИКОВА Тамара Миколаївна

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

**«28» лютого 2026 р.**

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: **ГОЛОВНЯ Євген Миколайович**

Керівник роботи: **ЛОБАНЧИКОВА Надія Миколаївна**

Тема роботи: **«Вебплатформа для онлайн-навчання»**,

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

Вихідні дані для роботи: об'єктом дослідження є процеси створення програмного забезпечення для освітніх потреб; предметом дослідження є методи та технології розробки вебплатформи для онлайн-навчання із використанням сучасних JavaScript-фреймворків та бібліотек на рівні фронтенду (React, TypeScript) і серверної частини (NestJS, TypeORM, PostgreSQL).

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

| Розділ | Консультант      | Завдання видав | Завдання прийняв |
|--------|------------------|----------------|------------------|
| 1      | ЛОБАНЧИКОВА Н.М. | 01.03.2026р.   | 01.03.2026р.     |
| 2      | ЛОБАНЧИКОВА Н.М. | 01.04.2026р.   | 01.04.2026р.     |
| 3      | ЛОБАНЧИКОВА Н.М. | 01.05.2026р.   | 01.05.2026р.     |

## РЕФЕРАТ

У роботі розглянуто розробку веб-платформи для організації онлайн-навчання з можливістю перегляду курсів, проходження тестування та здійснення оплати навчальних матеріалів. Метою роботи є створення сучасної інформаційної системи, що об'єднує навчальний контент, перевірку знань та комерційні функції в єдиному середовищі.

Об'єктом дослідження є процес взаємодії користувача з онлайн-освітнім середовищем, а предметом - методи та засоби розробки веб-застосунків на основі сучасних JavaScript-технологій.

Під час виконання роботи було проаналізовано існуючі освітні сервіси та виявлено проблему фрагментації навчального процесу між різними платформами. Для реалізації системи обрано стек технологій React, TypeScript, NestJS та PostgreSQL. Авторизацію користувачів реалізовано засобами JWT-автентифікації з підтримкою OAuth, взаємодію з базою даних - через ORM TypeORM, а обробку платежів – через платіжний сервіс WayForPay. Спроектовано структуру бази даних PostgreSQL, створено модулі курсів, уроків, тестів і оплат, а також реалізовано адаптивний інтерфейс користувача. Розгортання системи забезпечується за допомогою Docker.

Повний обсяг роботи становить 101 сторінка основного тексту, 9 таблиць, 23 ілюстрації, 1 додаток, 34 літературні джерела.

Результатом роботи є функціональний прототип освітньої платформи, що забезпечує централізоване керування навчальним процесом та спрощує доступ користувачів до освітнього контенту. Практична цінність полягає у можливості використання системи як основи для реального комерційного онлайн-сервісу навчання.

Ключові слова: ВЕБ-ПЛАТФОРМА, ОНЛАЙН-НАВЧАННЯ, REACT, NESTJS, TYPESCRIPT.

|           |      |                  |        |      |                                  |                  |      |        |
|-----------|------|------------------|--------|------|----------------------------------|------------------|------|--------|
|           |      |                  |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ         |                  |      |        |
| Змн.      | Арк. | № докум.         | Підпис | Дата | Вебплатформа для онлайн-навчання | Лім.             | Арк. | Аркуші |
| Розроб.   |      | Головня Є.М.     |        |      |                                  |                  |      |        |
| Перевір.  |      | Лобанчикова Н.М. |        |      |                                  |                  | 3    | 101    |
| Керівник  |      |                  |        |      |                                  | ФІКТ Гр. ВТ-22-1 |      |        |
| Н. контр. |      |                  |        |      |                                  |                  |      |        |
| Зав. каф. |      |                  |        |      |                                  |                  |      |        |

## ABSTRACT

This work presents the development of a web platform for organizing online learning, providing users with the ability to view courses, complete tests, and make payments for educational materials. The aim of the project is to create a modern information system that integrates educational content, knowledge assessment, and commercial functionality within a unified environment.

The object of the research is the process of user interaction with an online educational environment, while the subject of the study includes methods and tools for developing web applications based on modern JavaScript technologies.

During the research, existing e-learning platforms were analyzed, revealing the problem of fragmentation of the learning process across multiple systems. To implement the platform, a technology stack consisting of React, TypeScript, NestJS, and PostgreSQL was selected. User authentication was implemented using JWT-based authentication with OAuth support, database interaction was handled through the TypeORM ORM, and payment processing was integrated via the WayForPay payment service. A PostgreSQL database structure was designed, including modules for courses, lessons, tests, and payments, as well as a responsive user interface. System deployment is managed with Docker.

The total volume of the work is 101 pages of main text, 9 tables, 23 illustrations, 1 appendix, and 34 references.

The result of the work is a functional prototype of an educational platform that provides centralized management of the learning process and simplifies user access to educational content. The practical value of the project lies in its potential use as a foundation for a real commercial e-learning service.

Keywords: WEB PLATFORM, ONLINE LEARNING, REACT, NESTJS, TYPESCRIPT.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 4    |

## ЗМІСТ

|                                                                                      |    |
|--------------------------------------------------------------------------------------|----|
| ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ .....                                                      | 6  |
| ВСТУП.....                                                                           | 7  |
| 1. АНАЛІЗ ВЕБПЛАТФОРМ ДЛЯ ОНЛАЙН-НАВЧАННЯ .....                                      | 10 |
| 1.1 Постановка задачі .....                                                          | 10 |
| 1.2 Аналіз аналогів програмного продукту.....                                        | 13 |
| 1.3 Вибір архітектури вебпорталу для онлайн-навчання.....                            | 17 |
| 1.4 Обґрунтування вибору технологій реалізації платформи .....                       | 22 |
| Висновки до першого розділу .....                                                    | 25 |
| РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ОНЛАЙН-<br>НАВЧАННЯ.....           | 27 |
| 2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи ... | 27 |
| 2.2 Розробка моделей даних та взаємодії з API.....                                   | 35 |
| 2.3 Проєктування та реалізація алгоритмів роботи системи.....                        | 43 |
| 2.4 Реалізація функціоналу системи вебплатформи для онлайн-навчання .....            | 52 |
| Висновки до другого розділу .....                                                    | 60 |
| РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ОНЛАЙН-<br>НАВЧАННЯ.....       | 63 |
| 3.1 Порядок встановлення та налаштування параметрів платформи.....                   | 63 |
| 3.2 Структура інтерфейсу вебплатформи онлайн-навчання .....                          | 69 |
| 3.3 Тестування платформи.....                                                        | 79 |
| Висновки до третього розділу .....                                                   | 84 |
| ВИСНОВКИ.....                                                                        | 85 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....                                                      | 88 |
| ДОДАТКИ.....                                                                         | 91 |

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 5    |

## ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – Application Programming Interface (програмний інтерфейс застосунків)  
CRA – Create React App (інструмент ініціалізації React-проекту)  
CSR – Client-Side Rendering (клієнтський рендеринг)  
DB – Database (база даних)  
Docker – Платформа для контейнеризації застосунків  
JWT – JSON Web Token (веб-токен доступу)  
LMS – Learning Management System (система управління навчанням)  
NestJS – Фреймворк для розробки серверних застосунків на Node.js  
Node.js – Середовище виконання JavaScript на сервері  
OAuth – Open Authorization (протокол авторизації)  
ORM – Object-Relational Mapping (об'єктно-реляційне відображення)  
PostgreSQL – Реляційна система керування базами даних  
React – Бібліотека для створення інтерфейсів  
REST – Representational State Transfer (архітектурний стиль API)  
SQL – Structured Query Language (структурована мова запитів)  
SSG – Static Site Generation (статична генерація сайту)  
SSR – Server-Side Rendering (серверний рендеринг)  
TypeORM – ORM-бібліотека для роботи з базами даних у TypeScript/Node.js  
TypeScript – Типізована надбудова над JavaScript  
UI – User Interface (користувацький інтерфейс)  
UI/UX – Проєктування інтерфейсу та досвіду користувача  
UX – User Experience (користувацький досвід)  
WayForPay – Платіжний сервіс для обробки онлайн-транзакцій  
Web Vitals – Метрики продуктивності вебзастосунків  
IC – Інформаційна система

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 6    |

## ВСТУП

**Актуальність теми.** Стрімкий розвиток інформаційних технологій та глобальна цифровізація освітньої сфери суттєво змінюють підходи до організації навчального процесу. Дистанційне навчання перетворилося з тимчасового заходу на повноцінну форму здобуття освіти, яка охоплює мільйони користувачів у всьому світі. Зростаючий попит на онлайн-курси, зумовлений потребою у безперервному навчанні та підвищенні кваліфікації, підкреслює важливість розробки сучасних освітніх платформ. Водночас аналіз наявних рішень, зокрема Moodle, Coursera, Udemy та Google Classroom, виявляє суттєву проблему – фрагментацію функціонального забезпечення: навчальний контент, системи оцінювання, комунікаційні інструменти та механізми монетизації реалізуються як розрізнені сервіси, що ускладнює користувацький досвід і знижує ефективність навчального процесу. У цьому контексті розробка єдиної інтегрованої вебплатформи, яка об'єднує весь цикл дистанційного навчання від створення курсів до оцінювання результатів і комерціалізації освітнього контенту, – є актуальним і практично значущим завданням.

Наявні дослідження у сфері веб-орієнтованих систем управління навчанням (LMS) та сучасного стеку JavaScript-технологій підтверджують ефективність поєднання компонентного підходу на клієнтській частині з модульною серверною архітектурою для побудови масштабованих освітніх рішень. Використання типізованих мов програмування та ORM-інструментів для роботи з реляційними базами даних забезпечує надійність і підтримуваність програмних систем цього класу.

**Мета роботи** – розробка та програмна реалізація повнофункціональної вебплатформи для онлайн-навчання, що забезпечує централізоване управління навчальним контентом, відстеження прогресу студентів, перевірку знань, обробку платежів та адміністрування системи в межах єдиного програмного комплексу.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 7    |

**Завдання роботи.** Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної області та існуючих програмних аналогів освітніх платформ з метою виявлення їх функціональних можливостей та обмежень;
- визначити архітектуру та узагальнену структуру майбутнього програмного комплексу;
- обґрунтувати та здійснити вибір засобів реалізації системи;
- спроектувати модель даних і визначити об'єктно-орієнтовану структуру системи;
- розробити та реалізувати алгоритми роботи ключових підсистем: автентифікації, управління навчальним процесом, обробки платіжних транзакцій та автоматичного формування сертифікатів;
- реалізувати програмний комплекс вебплатформи для онлайн-навчання із застосуванням сучасного стеку технологій React, TypeScript, NestJS та PostgreSQL;
- виконати тестування реалізованого програмного продукту та підтвердити коректність його роботи.

**Об'єктом дослідження** є процеси створення програмного забезпечення для освітніх потреб.

**Предметом дослідження** є методи та технології розробки вебплатформи для онлайн-навчання із використанням сучасних JavaScript-фреймворків та бібліотек на рівні фронтенду (React, TypeScript) і серверної частини (NestJS, TypeORM, PostgreSQL).

**Методи дослідження.** У роботі використано методи системного аналізу предметної області та існуючих програмних рішень; методи об'єктно-орієнтованого проектування для побудови структури програмного комплексу; методи компонентного та модульного проектування для організації клієнтської і серверної частин системи; методи реляційного моделювання для

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 8    |

проектування схеми бази даних; методи інтеграційного та end-to-end тестування для верифікації реалізованих функціональних модулів.

**Практичне значення одержаних результатів** полягає у тому, що розроблений програмний комплекс є повнофункціональним прототипом освітньої платформи, готовим до розгортання у виробничому середовищі. Система може використовуватися як основа для реального комерційного онлайн-сервісу навчання, а також як базова інфраструктура для корпоративного навчання персоналу, університетських дистанційних курсів або авторських освітніх продуктів. Реалізована платформа розгорнута у хмарному середовищі Render та доступна для тестування за публічним посиланням.

**Структура та обсяг роботи.** Кваліфікаційна робота складається з вступу, 3 розділів, висновків, списку використаних джерел із 34 найменувань та додатків з лістингами програмного коду. Повний обсяг роботи становить 101 сторінка основного тексту. Робота містить таблиці та ілюстрації (UML-діаграми, скриншоти інтерфейсу, діаграми активності та архітектурні схеми) по тексту.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 9    |

# 1. АНАЛІЗ ВЕБПЛАТФОРМ ДЛЯ ОНЛАЙН-НАВЧАННЯ

## 1.1 Постановка задачі

Вебплатформа для онлайн-навчання є складною інформаційною системою, яка поєднує в собі широкий спектр функціональних можливостей, спрямованих на забезпечення ефективного та зручного процесу дистанційної освіти. У сучасних умовах цифровізації освітньої сфери такі системи набувають особливої значущості, оскільки вони дозволяють організувати навчальний процес незалежно від географічного розташування користувачів, забезпечуючи при цьому високий рівень доступності освітнього контенту. Зростаючий попит на дистанційне навчання, зумовлений як глобальними тенденціями, так і потребами сучасного ринку праці, підкреслює актуальність розробки подібних рішень.

Подібні платформи інтегрують у собі не лише навчальні матеріали, але й інструменти взаємодії між учасниками освітнього процесу, механізми оцінювання знань, а також засоби комерціалізації освітніх продуктів. Таким чином, вебплатформа виступає як єдина цифрова екосистема, що підтримує повний цикл навчальної діяльності - від створення контенту до його споживання та оцінювання результатів. Подібний підхід дозволяє уникнути розрізненості інструментів і забезпечити цілісний досвід як для студентів, так і для викладачів.

Архітектурно система складається з декількох взаємопов'язаних модулів, кожен з яких виконує окрему функціональну роль. Центральним елементом є модуль управління навчальним контентом, який відповідає за створення, структурування та оновлення курсів. Саме цей модуль забезпечує можливість формування навчальних програм, поділених на логічні блоки, теми та окремі уроки, що містять різноманітні навчальні матеріали, включаючи відео, презентації та текстові ресурси. Гнучка структура контенту дозволяє викладачам адаптувати матеріали під різні рівні підготовки аудиторії.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 10   |

Не менш важливим є модуль управління користувачами, який реалізує функціонал реєстрації, автентифікації та розмежування прав доступу. Він дозволяє визначати ролі користувачів, зокрема студентів, викладачів та адміністраторів, а також забезпечує збереження історії їхньої активності в системі. Це створює основу для персоналізованого навчального досвіду та дозволяє кожному користувачу отримувати лише ту інформацію, що відповідає його ролі та рівню доступу.

Модуль навчального процесу відповідає за безпосереднє проходження курсів, контроль прогресу та відкриття нових навчальних матеріалів відповідно до логіки навчальної програми. Послідовне розблокування тем мотивує студентів до систематичного опанування матеріалу. Додатково передбачається можливість автоматичного формування сертифікатів після успішного завершення курсів, що підвищує мотиваційний аспект навчання та надає підтвердження здобутих компетентностей.

Окрему роль відіграє модуль тестування, який забезпечує створення перевірочних завдань різних типів, автоматичну перевірку відповідей та формування статистики результатів. Це дозволяє об'єктивно оцінювати рівень засвоєння матеріалу та надавати зворотний зв'язок студентам у режимі реального часу.

Також у системі передбачається модуль електронної комерції, який відповідає за реалізацію фінансових операцій, включаючи продаж курсів, управління підписками та облік платежів. Даний модуль є важливою складовою, оскільки дозволяє забезпечити економічну модель функціонування платформи та відкриває можливості для монетизації освітнього контенту викладачами.

Основним етапом реалізації системи є проєктування інтерфейсу користувача із застосуванням сучасних підходів UI/UX, що дозволяють створити інтуїтивно зрозуміле та адаптивне середовище взаємодії. Якісний інтерфейс безпосередньо впливає на залученість користувачів та ефективність навчального процесу. Для цього використовуються можливості бібліотеки

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 11   |

React у поєднанні з сучасними підходами до побудови компонентного інтерфейсу, що забезпечує гнучкість у розробці інтерфейсних елементів[1].

Наступним важливим етапом є побудова серверної архітектури на базі NestJS із модульною організацією бізнес-логіки, що дозволяє ефективно розподіляти відповідальність між різними частинами серверного застосунку [2]. Взаємодія з базою даних здійснюється через ORM TypeORM, що забезпечує типобезпечний доступ до даних та підвищує надійність системи [5].

З метою забезпечення безпеки користувачів передбачається реалізація JWT-автентифікації з підтримкою OAuth-провайдерів [13]. Для реалізації платіжної функціональності інтегрується платіжний сервіс WayForPay, який забезпечує безпечну обробку транзакцій відповідно до вимог вітчизняного ринку [7].

Бізнес-логіка системи реалізується у вигляді REST API на базі NestJS, а валідація вхідних даних забезпечується засобами серверної частини. Для управління локальним станом клієнтської частини використовується вбудований механізм стану React, що дозволяє ефективно контролювати такі дані, як прогрес навчання та стан кошика, забезпечуючи узгодженість інтерфейсу під час переходів між сторінками.

Завершальним етапом є комплексне тестування системи, яке включає перевірку коректності фінансових операцій, функціональності навчального процесу, системи оцінювання, а також адаптивності інтерфейсу та продуктивності. Тестування дозволяє виявити та усунути потенційні вразливості до введення системи в експлуатацію.

У результаті реалізації всіх зазначених етапів створюється повноцінна вебплатформа, яка забезпечує ефективну взаємодію між усіма учасниками освітнього процесу та об'єднує навчальні, адміністративні та комерційні функції в межах єдиної системи.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 12   |

## 1.2 Аналіз аналогів програмного продукту

Розробка вебплатформи для онлайн-навчання потребує ретельного аналізу існуючих рішень на ринку з метою виявлення їх переваг та недоліків. У межах дослідження було проведено порівняльний огляд популярних освітніх платформ дистанційного навчання, що дозволило сформулювати чітке розуміння поточного стану галузі.

Однією з найвідоміших платформ є Moodle. Вона надає широкий функціонал управління курсами, можливість створення модулів навчання, тестів, завдань та системи оцінювання. Перевагою є гнучкість налаштування, велика кількість плагінів та можливість адаптації під потреби різних освітніх установ. Платформа підтримує журнал оцінок, ролі користувачів та структуровані навчальні програми. Недоліком є застарілий інтерфейс, перевантажена навігація та складність освоєння для нових користувачів. Значна кількість функцій реалізується через сторонні модулі, що призводить до різноманітності UX та нестабільності роботи після оновлень [33].

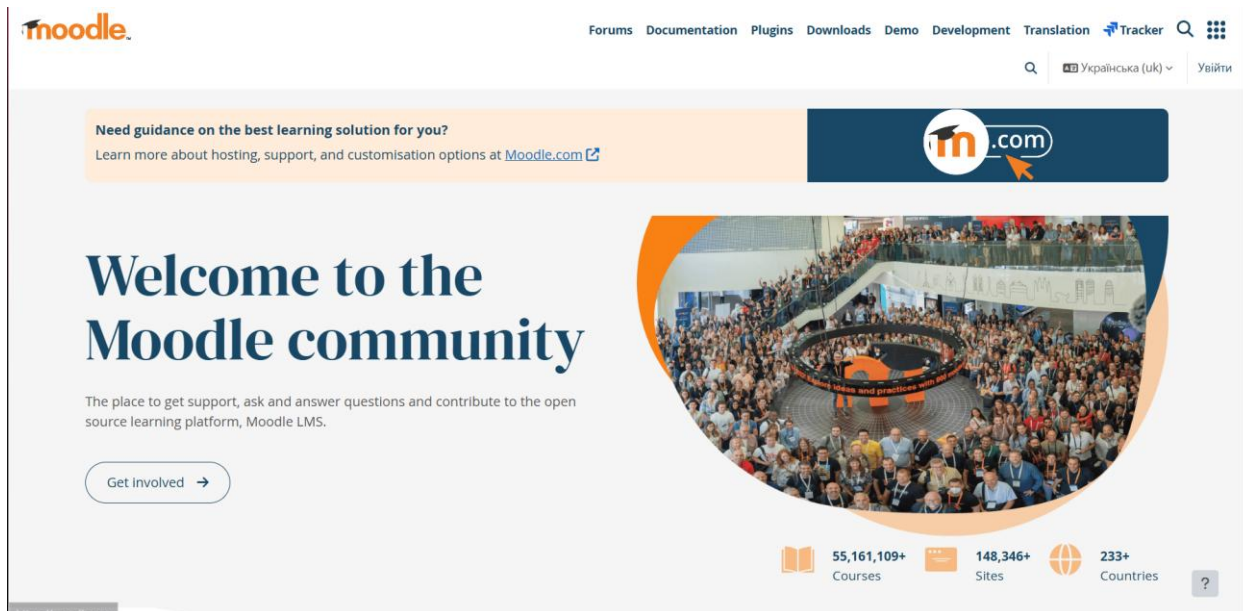


Рис. 1.1 – Вебпортал moodle

Наступним аналогом є Coursera. Платформа відзначається якісним дизайном, чіткою структурою курсів, відеолекціями та автоматизованими тестами. Реалізовано систему сертифікації, рекомендації курсів та персоналізацію навчання. Важливою перевагою є продуманий навчальний

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 13   |

сценарій із поступовим відкриттям матеріалів та мотиваційними механіками. Недоліком є обмежений доступ до редагування курсів для викладачів поза партнерськими програмами, а також відсутність повної гнучкості структури навчання. Частина функціоналу доступна лише в межах платної підписки, що зменшує доступність платформи[16].

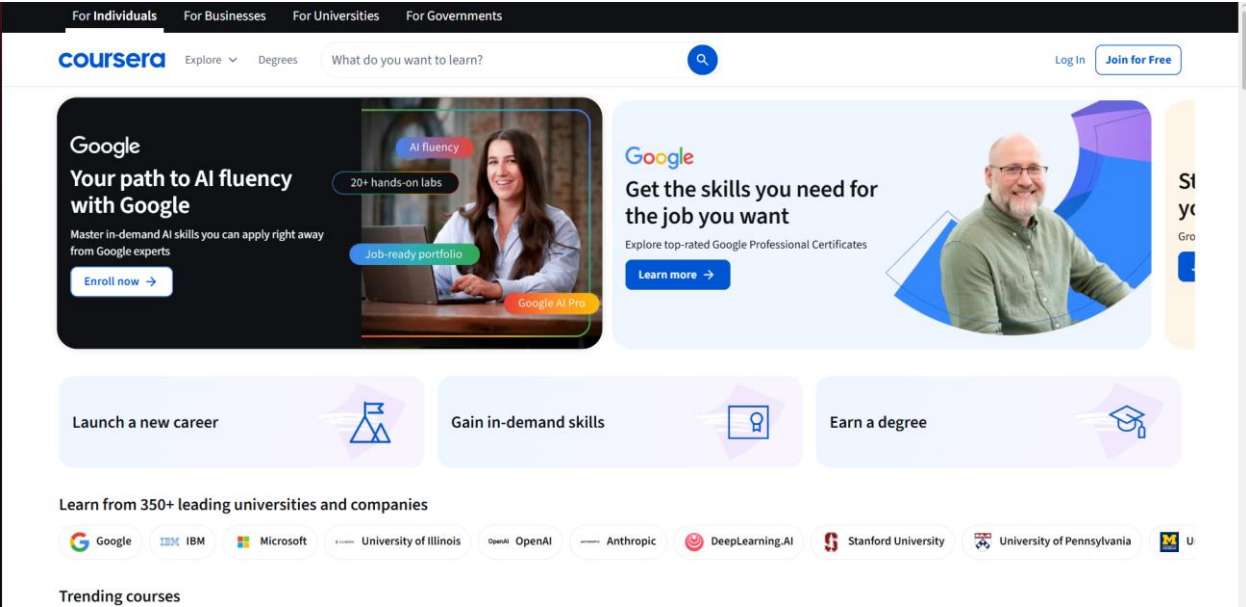


Рис. 1.2 – Вебпортал coursera

Ще одним поширеним рішенням є Udey. Платформа дозволяє будь-якому автору створювати власні курси, додавати відео, матеріали та тести. Система продажу курсів інтегрована безпосередньо в платформу, присутні відгуки користувачів і рейтингова система. Основною перевагою є простота створення контенту та зручний інтерфейс для студентів. Однак відсутня чітка стандартизація якості курсів, структура навчання часто залежить лише від автора, а аналітика успішності обмежена. Крім того, інструменти контролю прогресу та навчальної статистики значно простіші, ніж у академічних LMS[34].

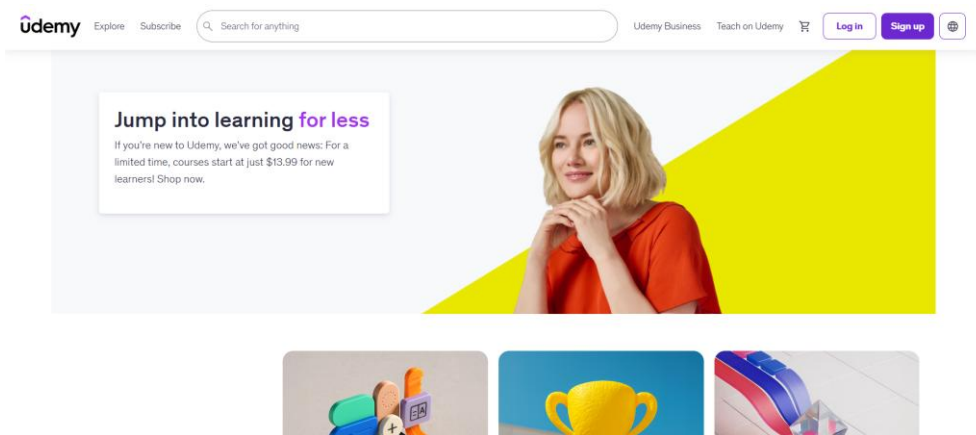


Рис. 1.3 – Вебпортал udemy

Google Classroom використовується переважно в навчальних закладах і забезпечує базову взаємодію між викладачем та студентами. Система добре інтегрована з хмарними сервісами, дозволяє видавати завдання, перевіряти роботи та комунікувати в межах курсу. Перевагою є простота використання та швидке розгортання. Недоліком є відсутність повноцінної комерційної моделі, обмежений функціонал тестування, відсутність складної аналітики прогресу та слабка кастомізація структури навчальних програм.

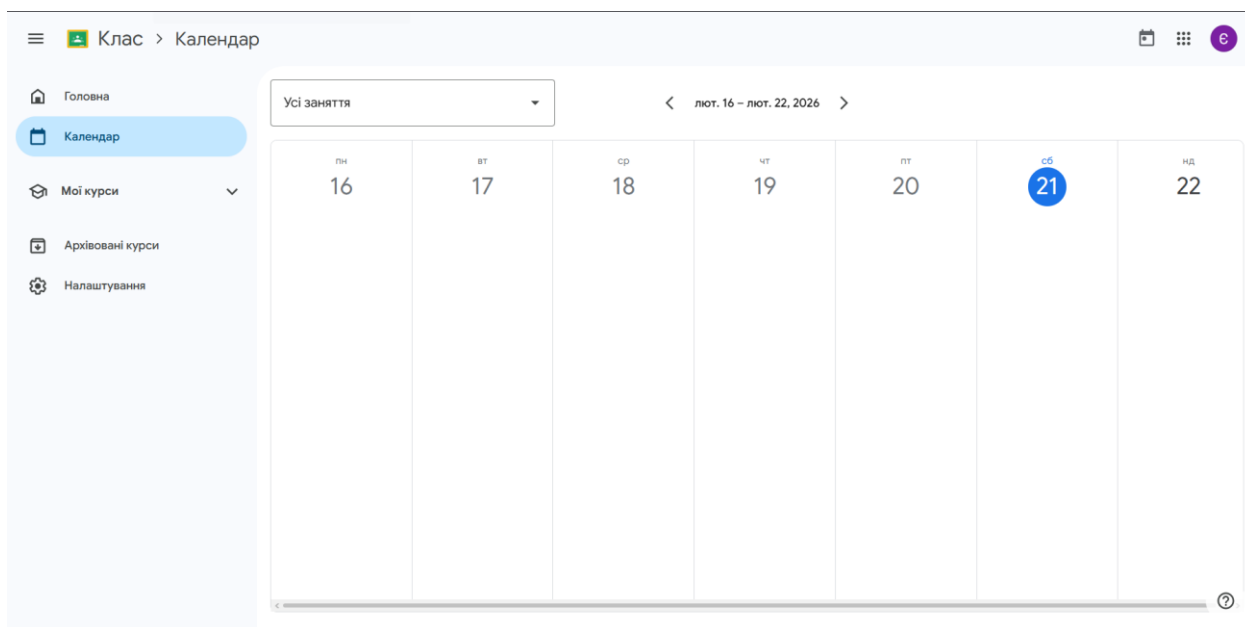


Рис. 1.4 – Вебпортал Google Classroom

Узагальнюючи аналіз існуючих платформ, можна зробити висновок, що більшість сучасних систем мають спільну проблему фрагментації функціоналу. Часто навчальний контент, оплата курсів, відеохостинг та комунікація реалізовані як окремі сервіси або зовнішні інтеграції, що порушує

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 15   |

цілісність користувацького досвіду. Також спостерігаються недоліки в аналітиці прогресу, персоналізації навчання або зручності створення курсів для викладачів.

Узагальнюючи аналіз існуючих вебпорталів, складемо порівняльну таблицю їх функціональних можливостей та нашого проєкту (таблиця 1.1).

Таблиця 1.1

Порівняння платформ онлайн-навчання

| Характеристика                                            | Coursera | Moodle | Google Classroom | Udemy | Розроблювана платформа |
|-----------------------------------------------------------|----------|--------|------------------|-------|------------------------|
| Сучасний технологічний стек                               | +        | +-     | +-               | +     | +                      |
| Єдина інтегрована система навчання                        | +-       | +      | +-               | +-    | +                      |
| Контекстні рекомендації курсів                            | +        | -      | -                | +     | +                      |
| Деталізована статистика прогресу                          | +        | +      | +-               | +-    | +                      |
| Керування навчальною траєкторією                          | +-       | +      | +-               | -     | +                      |
| Єдина система оплати всередині платформи                  | -        | -      | -                | -     | +                      |
| Глобальний пошук по матеріалах                            | +        | +-     | -                | +     | +                      |
| Інтеграція відео, тестів і матеріалів в одному середовищі | +-       | +      | +-               | +-    | +                      |

Розроблювана платформа має поєднати найкращі практики розглянутих сервісів, але усунути їх критичні проблеми UX та архітектури, створивши

єдину освітню екосистему. На відміну від існуючих рішень, що часто базуються на монолітних CMS або наборі сторонніх сервісів, система використовуватиме сучасний стек React та TypeScript на клієнті і NestJS на сервері, що забезпечить високу продуктивність і швидке завантаження сторінок. Платформа матиме інтегровану систему продажу курсів із єдиним кошиком і спільною сесією користувача на всіх сторінках, розширену аналітику прогресу навчання, глобальний пошук по всьому освітньому контенту та адаптивний інтерфейс.

Використання PostgreSQL та TypeORM дозволить реалізувати збереження історії навчання і статистики проходження курсів, автоматичне формування сертифікатів і архівацію попередніх результатів. Оптимізована доставка відеоматеріалів через хмарні сервіси усуне затримки відтворення, а централізована система управління курсами забезпечить однаковий досвід для студентів і викладачів. Таким чином, платформа стане цілісним середовищем дистанційного навчання, що поєднує створення контенту, навчання, оцінювання та оплату в межах єдиної системи.

### 1.3 Вибір архітектури вебпорталу для онлайн-навчання

Для реалізації вебплатформи онлайн-навчання обрано клієнт-серверну архітектуру[11]. Такий вибір зумовлений необхідністю поєднання високої продуктивності під час відображення освітнього контенту, що важливо для швидкого доступу до навчальних матеріалів і SEO-індексації курсів, а також високої інтерактивності для проходження уроків, тестів і здійснення оплат. Поєднання цих двох вимог в межах однієї архітектурної моделі є одним із ключових переваг обраного підходу. Система проєктується як модульний моноліт, де, попри розгортання як єдиного застосунку, внутрішня структура поділена на ізольовані бізнес-модулі: управління курсами, навчальний процес, тестування, користувачі та електронна комерція. Такий підхід дозволяє реалізувати концепцію Full-Stack додатку в межах єдиної кодової бази та забезпечити тісну інтеграцію клієнтської і серверної логіки.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 17   |

Вибір технологічного стеку базується на вимогах до швидкодії та масштабованості. Клієнтська частина реалізована як SPA на базі React 18 з TypeScript, зібрана через Create React App. Маршрутизація забезпечується бібліотекою React Router DOM. Завдяки SPA-архітектурі навігація між сторінками відбувається без перезавантаження, що забезпечує швидкий і безперервний користувацький досвід.

Оскільки система працює зі структурованими даними зі складними зв'язками, такими як взаємодія між курсами, уроками, тестами та результатами користувачів, як сховище даних обрано реляційну систему управління базами даних PostgreSQL[4]. Взаємодія з базою даних здійснюється через TypeORM, яка виступає прошарком доступу до даних. Перевагою такого рішення є повна типобезпека завдяки інтеграції з мовою TypeScript, що мінімізує помилки під час розробки та гарантує цілісність даних при виконанні транзакцій.

Для забезпечення надійності інтерфейсу впроваджено сучасні підходи до управління станом і валідації даних. Глобальний клієнтський стан, зокрема прогрес проходження курсу, відкриті уроки та кошик покупок, керується вбудованими механізми стану React, що дозволяє зберігати дані під час переходу між сторінками без їх втрати.

Для зменшення навантаження на основний сервер частина функціоналу делегується спеціалізованим хмарним сервісам. Зберігання та доставка відеолекцій і навчальних матеріалів здійснюється через UploadThing, що пришвидшує відтворення медіаконтенту. Обробка фінансових транзакцій при придбанні курсів виконується через платіжний шлюз WayForPay [7], який забезпечує безпечну обробку платежів, а система автентифікації реалізована за допомогою JWT із підтримкою OAuth-провайдерів, що дозволяє користувачам входити через наявні облікові записи сторонніх сервісів.

Архітектурна модель передбачає розділення системи на логічні зони доступу. Публічна частина, що включає каталог курсів і опис програм навчання, використовує переважно статичну генерацію сторінок для максимальної швидкості. Особистий кабінет користувача працює з

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 18   |

динамічним рендерингом для відображення персонального прогресу, результатів тестів і придбаних курсів. Адміністративна панель захищена системою ролей і надає інструменти управління курсами, користувачами та навчальним процесом. Таке розмежування забезпечує безпеку, стабільність і зручність використання платформи для всіх категорій користувачів.

Важливим аспектом обраної архітектури є принцип розмежування відповідальності між клієнтською та серверною частинами системи. Клієнтська частина, реалізована як Single Page Application на базі React, несе відповідальність виключно за відображення даних та обробку взаємодій користувача. Уся бізнес-логіка, валідація даних і управління доступом зосереджені на серверній частині, що є ключовою вимогою безпеки для освітньої платформи з комерційною складовою. Такий підхід виключає можливість обходу обмежень доступу шляхом модифікації клієнтського коду, оскільки авторизація завжди перевіряється незалежно на сервері.

Комунікація між клієнтом і сервером здійснюється через REST API з використанням протоколу HTTP/HTTPS. Кожен запит до захищених ресурсів супроводжується JWT access-токеном у заголовку Authorization, що дозволяє серверу ідентифікувати користувача та перевірити його права доступу без звернення до бази даних для кожного запиту. Така схема є значно ефективнішою порівняно із сесійною автентифікацією, оскільки не вимагає зберігання стану сесій на сервері і добре масштабується при збільшенні кількості одночасних користувачів. Для операцій, що потребують підвищеного рівня безпеки, зокрема зміна пароля чи управління правами доступу, передбачено додаткову перевірку ідентичності.

Серверна частина реалізована на фреймворку NestJS із модульною організацією коду. Кожен функціональний домен системи - автентифікація, управління курсами, навчальний процес, тестування, платежі - інкапсульований у відповідному модулі з чітко визначеним публічним інтерфейсом. Міжмодульна взаємодія здійснюється виключно через оголошені сервіси, що запобігає виникненню прихованих залежностей і

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 19   |

спрощує тестування окремих компонентів у ізоляції. NestJS використовує концепцію декораторів та впровадження залежностей (Dependency Injection), що забезпечує низьку зв'язність компонентів і підвищує тестованість системи в цілому [2].

Схема бази даних спроектована відповідно до третьої нормальної форми, що забезпечує відсутність надлишкових даних і цілісність зв'язків між сутностями. Для кожної таблиці визначено первинні ключі у форматі UUID, що виключає виникнення колізій при розподіленому генеруванні ідентифікаторів та унеможлиблює передбачення значень ключів зловмисниками, на відміну від послідовних числових ідентифікаторів. Зовнішні ключі забезпечують каскадне видалення залежних записів там, де це відповідає бізнес-логіці: наприклад, видалення модуля курсу автоматично видаляє всі уроки та пов'язані записи прогресу. Унікальні складені індекси на таблицях Enrollment та Progress гарантують відсутність дублікатів записів навіть при паралельних запитах.

Архітектура системи передбачає чіткий поділ на шари представлення, бізнес-логіки та доступу до даних. На рівні представлення контролери NestJS приймають HTTP-запити, виконують попередню валідацію вхідних даних за допомогою DTO-класів і class-validator та делегують обробку відповідним сервісам. Сервісний шар реалізує всю бізнес-логіку: перевірку прав доступу, оркестрацію операцій над кількома сутностями, формування відповідей. Шар доступу до даних представлений репозиторіями TypeORM, які абстрагують деталі взаємодії з PostgreSQL. Такий трирівневий поділ дозволяє вносити зміни в реалізацію одного шару без впливу на інші, що суттєво прискорює розробку та підтримку системи [9].

Окремої уваги заслуговує підхід до організації клієнтського стану. В умовах SPA-архітектури управління станом є одним із ключових архітектурних рішень, оскільки від нього безпосередньо залежать продуктивність і передбачуваність поведінки інтерфейсу. Для даної платформи обрано React Context API як вбудований механізм управління

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 20   |

глобальним станом без залучення сторонніх бібліотек. Стан розбито на чотири незалежні контексти: автентифікація, тема оформлення, список бажаних і сповіщення. Такий поділ дозволяє компонентам підписуватися лише на необхідні їм дані та уникати зайвих ре-рендерів при зміні непов'язаних частин стану. Локальний стан окремих сторінок і компонентів управляється через хуки `useState` та `useReducer` безпосередньо всередині відповідних компонентів.

Безпека передачі даних між усіма компонентами системи забезпечується протоколом HTTPS з примусовим перенаправленням з HTTP. Паролі користувачів зберігаються виключно у вигляді хешів `bcrypt` із коефіцієнтом складності 10, що унеможливорює відновлення оригінальних паролів навіть при отриманні зловмисником прямого доступу до бази даних. Захист від атак типу CSRF забезпечується специфікою JWT-автентифікації: токен зберігається у `localStorage` і явно передається у заголовок кожного запиту, що унеможливорює його автоматичне надсилання браузером у складі cookie. Захист від SQL-ін'єкцій гарантується використанням параметризованих запитів через `TypeORM`, яка ніколи не включає призначені для користувача дані безпосередньо у текст SQL-запиту.

Для забезпечення стабільного середовища розробки та відтворюваних умов розгортання систему повністю контейнеризовано засобами `Docker`. Кожен компонент платформи - клієнтська частина, серверна частина та база даних - розміщується у власному ізольованому контейнері з чітко визначеними залежностями і конфігурацією. `Docker Compose` оркеструє запуск усіх контейнерів з правильним порядком ініціалізації та налаштуванням внутрішньої мережі. Такий підхід усуває проблему “на моїй машині це працює”: середовище розробки, тестування і виробничий стенд є ідентичними з точки зору конфігурації залежностей та версій програмного забезпечення [6]. Це значно прискорює процес введення нових розробників у проєкт та спрощує розгортання нових середовищ.

Обрана архітектура також враховує вимоги до масштабованості системи при зростанні кількості користувачів і навчального контенту. Модульна

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 21   |

організація серверної частини дозволяє у майбутньому виокремити окремі модулі у незалежні мікросервіси без переписування бізнес-логіки - достатньо змінити спосіб їхнього розгортання і комунікації. Зберігання медіаконтенту на зовнішніх хмарних сервісах знімає навантаження з основного сервера і дозволяє незалежно масштабувати сховище та обчислювальні ресурси. Реляційна база даних PostgreSQL підтримує горизонтальне масштабування через механізми реплікації, що уможлиблює розподіл навантаження читання між кількома вузлами при необхідності. Таким чином, поточна архітектурна модель є як придатною для навчального демонстраційного стенду, так і такою, що не потребує кардинального перепроєктування при переході до промислової експлуатації з реальним навантаженням.

#### 1.4 Обґрунтування вибору технологій реалізації платформи

Реалізація вебплатформи онлайн-навчання, що була сформульована у попередніх розділах, потребує використання сучасного технологічного стеку, здатного забезпечити комплексну роботу з навчальним контентом, користувачами, тестуванням знань, аналітикою прогресу та елементами електронної комерції. Як було визначено в постановці задачі, система повинна об'єднувати функції створення та управління курсами, організацію навчального процесу, перевірку знань студентів, збереження результатів навчання, а також підтримку монетизації через продаж курсів та підписок. Таким чином, ключовою вимогою до архітектури є створення єдиної інтегрованої платформи, яка замінює розрізнені сервіси, характерні для існуючих рішень.

Клієнтська частина системи реалізується за допомогою бібліотеки React, яка дозволяє будувати динамічні інтерфейси на основі компонентного підходу. Це є особливо важливим для освітньої платформи, оскільки користувач взаємодіє з великою кількістю різних сценаріїв у межах одного середовища, включаючи перегляд навчальних матеріалів, проходження уроків, виконання тестових завдань, перегляд результатів та роботу з

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 22   |

особистим профілем. Використання React забезпечує швидке оновлення окремих частин інтерфейсу без необхідності перезавантаження сторінки, що робить процес навчання більш безперервним і комфортним для користувача. Крім того, компонентна структура дозволяє повторно використовувати елементи інтерфейсу, що зменшує складність розробки та підвищує підтримуваність системи у довгостроковій перспективі.

Серверна частина реалізується на основі фреймворку NestJS, який побудований із використанням TypeScript і орієнтований на модульну архітектуру. Такий підхід наряду відповідає структурі системи, описаній у попередніх розділах, де платформа поділяється на логічні блоки, зокрема управління користувачами, навчальним контентом, процесом навчання, тестуванням та електронною комерцією. NestJS дозволяє організувати кожен із цих напрямів у вигляді окремих модулів із чітко визначеною відповідальністю, що значно спрощує масштабування системи та її подальший розвиток. Окрім цього, фреймворк забезпечує стандартизовану структуру серверної частини, що особливо важливо при розробці великих багатокомпонентних систем із залученням команди розробників.

Для збереження даних використовується PostgreSQL, яка є однією з найбільш надійних реляційних систем управління базами даних. У контексті даної платформи це рішення є доцільним через необхідність роботи зі складними взаємозв'язками між сутностями. Система повинна зберігати інформацію про користувачів різних ролей, структуру курсів, уроки, тести, результати проходження навчання, а також фінансові транзакції, пов'язані з покупкою курсів. PostgreSQL забезпечує високу цілісність даних, підтримку транзакцій та стабільну роботу при великому навантаженні, що є критичним для освітніх платформ, які можуть одночасно обслуговувати значну кількість користувачів.

Важливим елементом реалізації є використання TypeScript як єдиної мови програмування для клієнтської та серверної частин системи. Це дозволяє створити єдину типізовану модель даних, що значно зменшує кількість

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 23   |

помилки під час розробки та підвищує передбачуваність роботи системи. У межах складної предметної області, де взаємодіє велика кількість сутностей, типобезпека відіграє важливу роль у забезпеченні стабільності та надійності програмного продукту. Крім того, спільна мова для обох частин системи спрощує взаємодію між розробниками та пришвидшує процес розробки.

Архітектура системи побудована за клієнт-серверним принципом із модульною структурою серверної частини. Такий підхід дозволяє поєднати простоту монолітної системи з гнучкістю модульного поділу, що наближає архітектуру до мікросервісного стилю без надмірного ускладнення інфраструктури. Це є важливим у контексті освітньої платформи, яка повинна підтримувати різні типи користувачів, зокрема студентів, викладачів та адміністраторів, кожен із яких має власний набір функцій і рівень доступу до системи.

Для забезпечення стабільного середовища розробки та розгортання використовується Docker, який дозволяє контейнеризувати всі компоненти системи, включаючи клієнтську частину, серверну логіку та базу даних. Це гарантує однакову поведінку системи на всіх етапах життєвого циклу програмного продукту, починаючи від розробки і тестування та завершуючи продакшн-середовищем. Використання контейнеризації також спрощує масштабування системи при збільшенні навантаження та дозволяє швидко розгорнути нові середовища без конфліктів залежностей, що особливо важливо під час активного розвитку проєкту.

Таким чином, обраний технологічний стек у складі React, NestJS, PostgreSQL, TypeScript та Docker є обґрунтованим і повністю відповідає вимогам до сучасної вебплатформи онлайн-навчання. Він дозволяє реалізувати єдину цифрову екосистему, яка поєднує навчальний контент, систему тестування, управління користувачами та фінансові механізми в межах одного інтегрованого середовища, забезпечуючи при цьому високу продуктивність, надійність та можливість подальшого розширення функціоналу.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 24   |

## Висновки до першого розділу

У першому розділі було виконано комплексне дослідження предметної області та здійснено проєктування вебплатформи для онлайн-навчання, яка розглядається як сучасна інформаційна система освітнього типу. На початковому етапі було сформовано постановку задачі, у межах якої визначено основні вимоги до майбутньої системи. Зокрема, платформа повинна забезпечувати повний цикл дистанційного навчання, починаючи від створення та управління навчальним контентом і завершуючи проходженням курсів, тестуванням знань, формуванням результатів навчання та можливістю монетизації освітніх продуктів.

У процесі аналізу предметної області було встановлено, що сучасні освітні платформи відіграють важливу роль у цифровізації навчального процесу, оскільки забезпечують доступ до знань незалежно від географічного розташування користувача, а також дозволяють організовувати взаємодію між викладачами та студентами у зручному інтерактивному форматі. Окрему увагу було приділено функціональності, пов'язаній із контролем успішності, персоналізацією навчання та автоматизацією перевірки знань, що є невід'ємною складовою сучасних LMS-систем.

У межах порівняльного аналізу існуючих рішень, таких як Moodle, Coursera, Udemy та Google Classroom, було виявлено ряд спільних обмежень, характерних для більшості платформ. Найбільш суттєвою проблемою є фрагментація функціональних модулів, коли навчальний контент, тестування, комунікація та комерційні механізми реалізуються як окремі або недостатньо інтегровані сервіси. Це призводить до ускладнення користувацького досвіду, зниження цілісності системи та обмежує можливості подальшого масштабування. Саме тому було обґрунтовано необхідність створення єдиної інтегрованої платформи, яка об'єднує всі ключові процеси навчання в межах одного середовища.

На основі отриманих результатів було сформовано архітектурний підхід до реалізації системи. Обрана клієнт-серверна модель з модульною

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 25   |

структурою дозволяє ефективно розділити відповідальність між різними частинами системи та забезпечити її подальший розвиток без суттєвих змін у загальній структурі. Клієнтська частина реалізується за допомогою React, що забезпечує побудову динамічного та інтерактивного інтерфейсу, орієнтованого на зручну взаємодію користувача з навчальним контентом. Серверна частина побудована на основі NestJS, що дозволяє організувати бізнес-логіку у вигляді чітко структурованих модулів, які відповідають за різні аспекти системи, включаючи користувачів, курси, навчальний процес, тестування та електронну комерцію.

Значну увагу було приділено вибору системи зберігання даних, оскільки платформа працює зі складними структурованими зв'язками між сутностями. Використання PostgreSQL дозволяє забезпечити надійне збереження інформації про користувачів, курси, результати навчання та фінансові операції, а також гарантує цілісність даних завдяки підтримці транзакцій. У поєднанні з TypeScript це формує єдину типізовану модель даних, що підвищує стабільність системи та зменшує ймовірність виникнення помилок на етапі розробки.

Додатково було обґрунтовано використання Docker для контейнеризації всіх компонентів системи, що дозволяє забезпечити однакове середовище розробки, тестування та розгортання. Це значно спрощує процес впровадження системи та підвищує її масштабованість у разі збільшення навантаження або кількості користувачів.

Таким чином, у результаті виконання першого розділу було сформовано цілісну концепцію вебплатформи онлайн-навчання, визначено її функціональну структуру, архітектурні принципи та технологічну основу. Отримані результати створюють базу для подальшої програмної реалізації системи та переходу до наступних етапів розробки, включаючи детальне проєктування бази даних та реалізацію серверної і клієнтської частин.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 26   |

## РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ОНЛАЙН-НАВЧАННЯ

### 2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

На етапі проєктування вебплатформи онлайн-навчання першочерговим завданням є формалізація функціональних вимог до системи. Для цього застосовано методологію варіантів використання (Use Case), яка дозволяє описати взаємодію зовнішніх акторів із системою через набір сценаріїв. Паралельно розроблено об'єктно-орієнтовану модель, що відображає ключові сутності предметної області та їхні взаємозв'язки. Поєднання цих двох підходів забезпечує системність у прийнятті архітектурних рішень і слугує фундаментом для подальшого проєктування бази даних та реалізації серверної логіки.

Проєктування системи розпочинається з ретельного аналізу потреб кожної категорії користувачів. Освітня платформа принципово відрізняється від типового вебзастосунку тим, що одночасно обслуговує кілька принципово різних груп: тих, хто навчається, тих, хто навчає, та тих, хто адмініструє процес. Кожна з цих груп має власні сценарії взаємодії, власні очікування від системи та власні критерії якості роботи платформи. Саме тому детальне розмежування ролей і відповідних функцій є критично важливим кроком, що впливає на всі наступні рішення щодо архітектури та реалізації.

Відповідно до функціональної специфіки платформи, у системі виокремлено п'ять ролей акторів: студент, викладач, модератор, адміністратор і супер-адміністратор. Кожна роль характеризується власним набором дозволів і функціональних сценаріїв взаємодії із системою. Такий підхід до розмежування ролей ґрунтується на принципі найменших привілеїв, згідно з яким кожен користувач має доступ лише до тих функцій, що необхідні йому для виконання своїх завдань.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 27   |

Студент є основним кінцевим користувачем платформи та центральним актором усього навчального процесу. Він здійснює реєстрацію та автентифікацію в системі, у тому числі через Google OAuth, що значно спрощує процес входу. Після реєстрації студент отримує доступ до каталогу курсів із можливістю фільтрації за категорією, рівнем складності та ціновим діапазоном. Логіка доступу до навчального контенту організована таким чином, що безкоштовні уроки доступні без запису, тоді як платний або повний контент відкривається лише після оплати або запису на курс. Система відстежує прогрес навчання на рівні окремих уроків: фіксується кількість переглянутих секунд відео та статус завершення. Після проходження всіх уроків курсу студент може отримати PDF-сертифікат із унікальним кодом верифікації, що підтверджує факт успішного завершення навчання.

Викладач відповідає за наповнення платформи освітнім контентом і є ключовою фігурою з точки зору якості навчальних матеріалів. Функціонал викладача охоплює повний цикл роботи з курсом: від створення та налаштування до публікації та аналізу результатів. Викладач може формувати ієрархічну структуру навчальної програми через модулі та уроки, додавати матеріали різних типів - відеолекції, текстові матеріали у форматі Markdown та quiz-завдання з автоматичною перевіркою. Завантаження медіафайлів здійснюється через інтегровану форму з відображенням прогресу. Після завершення роботи над курсом викладач надсилає його на модерацію, за результатами якої адміністратор приймає рішення про публікацію. Також у межах функціоналу викладача передбачено перегляд аналітики успішності студентів, управління промокодами та можливість подання запиту на виплату зароблених коштів.

Адміністратор і супер-адміністратор здійснюють загальний нагляд і управління платформою на найвищому рівні. До їхніх повноважень належать: контроль облікових записів користувачів з можливістю блокування та розблокування, модерація курсів зі зміною статусу від DRAFT до PUBLISHED або ARCHIVED, управління промокодами та запитами на виплату, а також

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 28   |

моніторинг стану системи через журнал аудиту та розширену аналітику. Принципова відмінність між адміністратором і супер-адміністратором полягає в тому, що лише супер-адміністратор має право призначати нових адміністраторів. Модератор займає проміжне місце в ієрархії: він має обмежені права перегляду курсів і контенту без права адміністративних змін, що дозволяє залучати додаткових людей до процесу перевірки матеріалів без надання їм повного доступу до адмін-панелі.

На основі аналізу ролей визначено повний перелік варіантів використання, що охоплюють усі ключові сценарії взаємодії з платформою. Реєстрація та автентифікація користувачів реалізована у двох варіантах: через форму з email та паролем або через Google OAuth. Обидва варіанти підтримують механізм refresh-токенів для підтримки тривалих сесій, а також функцію скидання пароля через email з тимчасовим посиланням.

Перегляд каталогу курсів є одним із найбільш складних сценаріїв з точки зору фільтрації та сортування даних. Система підтримує одночасне застосування кількох фільтрів: пошук за текстом по назві та опису курсу, фільтрацію за категорією, рівнем складності, ціновим діапазоном та мінімальним рейтингом. Результати можна переглядати у двох режимах відображення - у вигляді сітки або списку, з підтримкою пагінації. Стан фільтрів синхронізується з URL-параметрами, що дозволяє зберігати та передавати посилання на відфільтровані результати.

Запис студента на курс відбувається за двома сценаріями залежно від типу курсу. Безкоштовні курси доступні для запису безпосередньо через кнопку на сторінці. Платні курси потребують проходження процедури оплати через платіжний шлюз WayForPay. Після успішного підтвердження транзакції система автоматично створює запис Enrollment та надає студенту доступ до всіх матеріалів курсу.

Проходження уроків реалізовано з урахуванням специфіки різних типів контенту. Для відеоуроків система відстежує прогрес перегляду в секундах і автоматично фіксує завершення уроку після перегляду більше 80% відео.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 29   |

Текстові матеріали позначаються як завершені через ручне підтвердження. Quiz-уроки передбачають відповіді на запитання з автоматичною перевіркою та відображенням результату; для зарахування уроку необхідно відповісти правильно більш ніж на 50% запитань.

Взаємодія з Q&A-системою дозволяє студентам ставити запитання до конкретного уроку та отримувати відповіді від викладача або інших студентів. Відповіді викладача позначаються спеціальним прапором isInstructor, що дозволяє одразу ідентифікувати авторитетну відповідь. Система відгуків забезпечує можливість залишати оцінку від 1 до 5 зірок з текстовим коментарем після завершення курсу. Відгуки проходять модерацію перед відображенням у публічному каталозі.

Обробка платіжних транзакцій через WayForPay реалізована з обов'язковою верифікацією підпису HMAC-MD5 для захисту від підробки повідомлень. Система підписок підтримує два плани - місячний та річний - з автоматичним встановленням терміну дії після успішної оплати. Промокоди надають можливість зменшити вартість курсу на визначений відсоток із обмеженням кількості використань. Автоматичне формування PDF-сертифікатів відбувається після підтвердження 100-відсоткового прогресу проходження курсу, а верифікація сертифіката за унікальним кодом доступна без автентифікації.

Управління вішлістом дозволяє авторизованим студентам зберігати курси в список бажаних з синхронізацією стану між сторінками. Реферальна система відстежує запрошення через унікальний токен, який генерується для кожного користувача і може бути поширений. Запит на виплату дозволяє викладачу ініціювати отримання зароблених коштів, а обробка цих запитів здійснюється адміністратором через адмін-панель. Журнал аудиту фіксує всі адміністративні дії з HTTP-методом, маршрутом та тілом запиту.

Діаграма варіантів використання, наведена на рис. 2.1, унаочнює межі системи та відображає функціональні зв'язки між акторами і відповідними сценаріями взаємодії.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 30   |

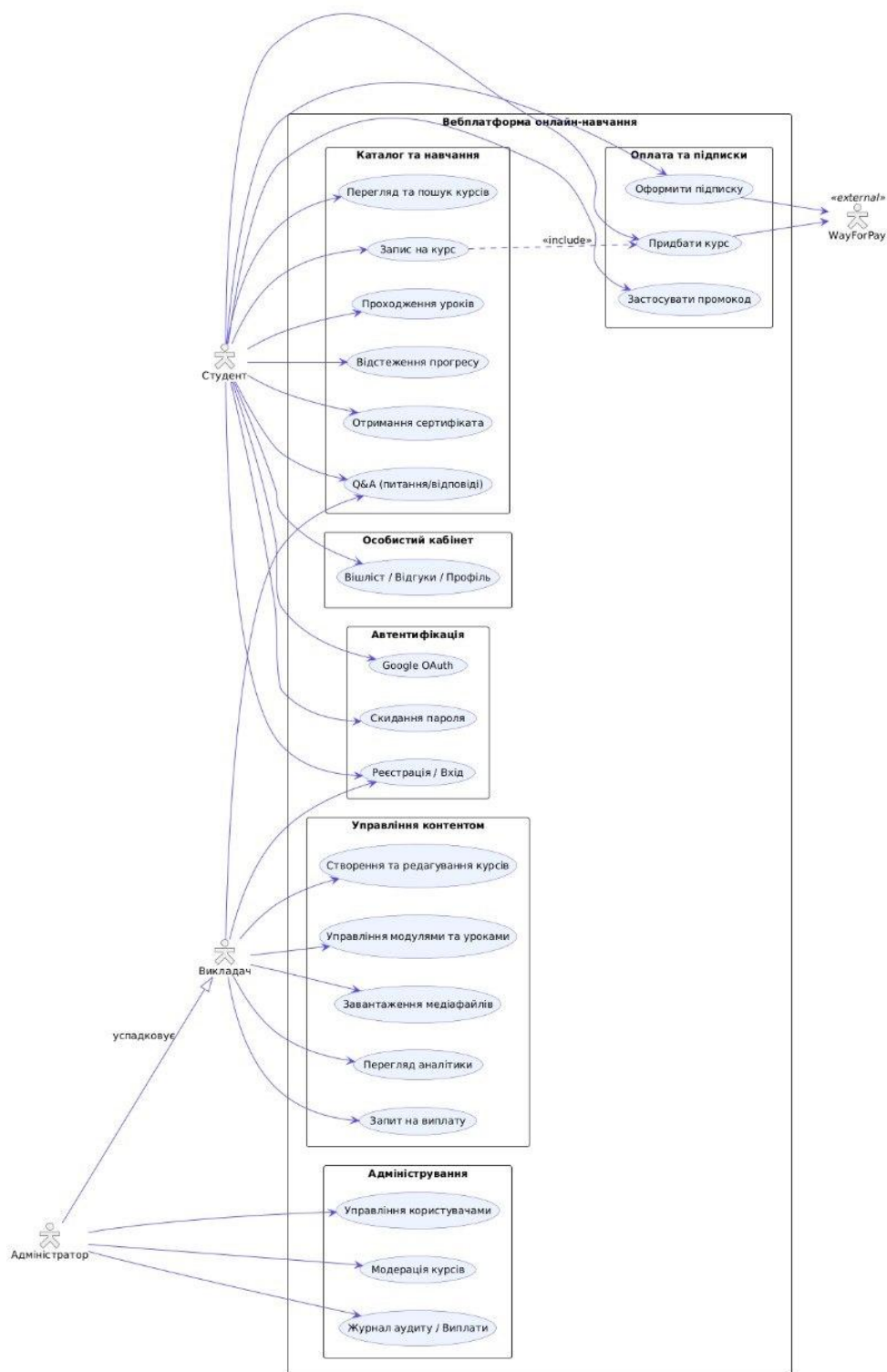


Рис. 2.1 – Діаграма варіантів використання системи

Об'єктно-орієнтована модель системи сформована на основі виявлених сутностей предметної області. Кожна сутність реалізується у вигляді окремої TypeORM-сутності (entity) з визначеними атрибутами та зв'язками. Модель

|      |      |                  |        |      |
|------|------|------------------|--------|------|
|      |      | Головня С.М.     |        |      |
|      |      | Лобанчикова Н.М. |        |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |

відображає реальну структуру навчального процесу і слугує безпосередньою основою для схеми бази даних.

Центральною сутністю системи є клас User, що зберігає загальну інформацію про користувача: ідентифікатор у форматі UUID, ім'я, електронну пошту, хеш пароля, посилання на аватар, роль у системі, прапор активності для механізму блокування, а також поля для зберігання причини та дати блокування. Для підтримки Google OAuth зберігається зовнішній ідентифікатор провайдера. Поля для скидання пароля містять хеш токена та час закінчення його дії. Клас User знаходиться в центрі майже всіх зв'язків системи: з ним пов'язані записи на курси, прогрес, платежі, сертифікати, підписки, відгуки та сповіщення.

Клас Course описує навчальний курс і містить: назву, опис, ціну, URL обкладинки, статус публікації, рівень складності, категорію та рейтинг. Курс пов'язаний із автором через зв'язок ManyToOne та агрегує колекцію модулів через OneToMany. Статус курсу визначає його видимість у каталозі: лише курси зі статусом PUBLISHED відображаються для студентів.

Клас CourseModule реалізує ієрархічну структуру курсу: кожен модуль містить назву та порядковий номер, а також агрегує уроки. Дворівнева структура "курс → модуль → урок" дозволяє гнучко організовувати навчальну програму, групуючи пов'язані теми в логічні блоки. Порядковий номер модуля забезпечує правильне відображення навчального плану.

Клас Lesson представляє окремий навчальний матеріал і є базовою одиницею навчального процесу. Урок містить заголовок, тип контенту, URL медіафайлу для відеоуроків, текстовий вміст у форматі Markdown для текстових матеріалів і зберігання JSON-структури питань для quiz-уроків, тривалість у секундах, порядковий номер та прапор безкоштовного перегляду. Наявність цього прапора дозволяє викладачу надавати безкоштовний доступ до окремих уроків для залучення нових студентів.

Клас Enrollment реалізує зв'язок між студентом і курсом. Унікальна пара userId та courseId гарантує відсутність дублікатів записів. Клас зберігає дату

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 32   |

запису та фактично сплачену ціну, що дозволяє коректно враховувати знижки від промокодів при аналізі доходів. Наявність цього запису визначає рівень доступу студента до матеріалів курсу.

Клас Progress відстежує прогрес студента на рівні окремого уроку та є ключовою сутністю для механізму контролю навчання. Він зберігає прапор завершення уроку, кількість переглянутих секунд відео та час останнього оновлення. Унікальна пара `userId` та `lessonId` гарантує відсутність дублікатів. Прогрес курсу обчислюється динамічно на основі агрегації записів Progress для всіх уроків курсу.

Клас Payment фіксує фінансову транзакцію: унікальний ідентифікатор замовлення, суму, статус обробки та посилання на курс і користувача. Статус платежу може набувати значень `pending`, `success` або `failure`. Запис Payment створюється при ініціації оплати і оновлюється за результатами `callback` від платіжного шлюзу.

Клас Certificate представляє PDF-сертифікат про завершення курсу. Він містить унікальний код верифікації, дату видачі та дані PDF у форматі Base64. Зберігання бінарних даних документа безпосередньо в базі даних забезпечує незалежність від зовнішніх файлових сховищ і гарантує збереження сертифіката навіть при зміні інфраструктури.

Клас Subscription описує підписку користувача на платформу: план, статус, сплачену ціну, ідентифікатор замовлення платіжного шлюзу, дату початку та закінчення терміну дії. Підписка надає студенту розширений доступ до контенту платформи на визначений термін.

Клас PromoCode містить унікальний код знижки, відсоток знижки, прив'язку до курсу або викладача, статус модерації, термін дії, ліміт та лічильник використань. Статус модерації дозволяє адміністратору контролювати промо-акції, що проводяться викладачами.

Клас Review зберігає відгук студента на курс: числовий рейтинг, текстовий коментар та прапор модерації. Унікальна пара `userId` та `courseId`

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 33   |

виключає дублювання відгуків. Рейтинг курсу перераховується автоматично при додаванні нового схваленого відгуку.

Класи QaQuestion та QaAnswer реалізують систему запитань і відповідей до уроків. Питання містить текст, посилання на урок та автора; відповідь - текст, посилання на питання, автора та прапор належності до викладача. Лічильник кількості відповідей оновлюється при додаванні нових відповідей для відображення активності дискусії.

Клас Notification зберігає системні сповіщення для користувача з типізованим полем типу та прапором прочитання. Клас AuditLog фіксує всі адміністративні дії з деталями запиту для забезпечення прозорості та безпеки. Клас Referral відстежує реферальні зв'язки між користувачами. Клас PayoutRequest фіксує запити викладачів на виплату зароблених коштів із відповідним статусом обробки.

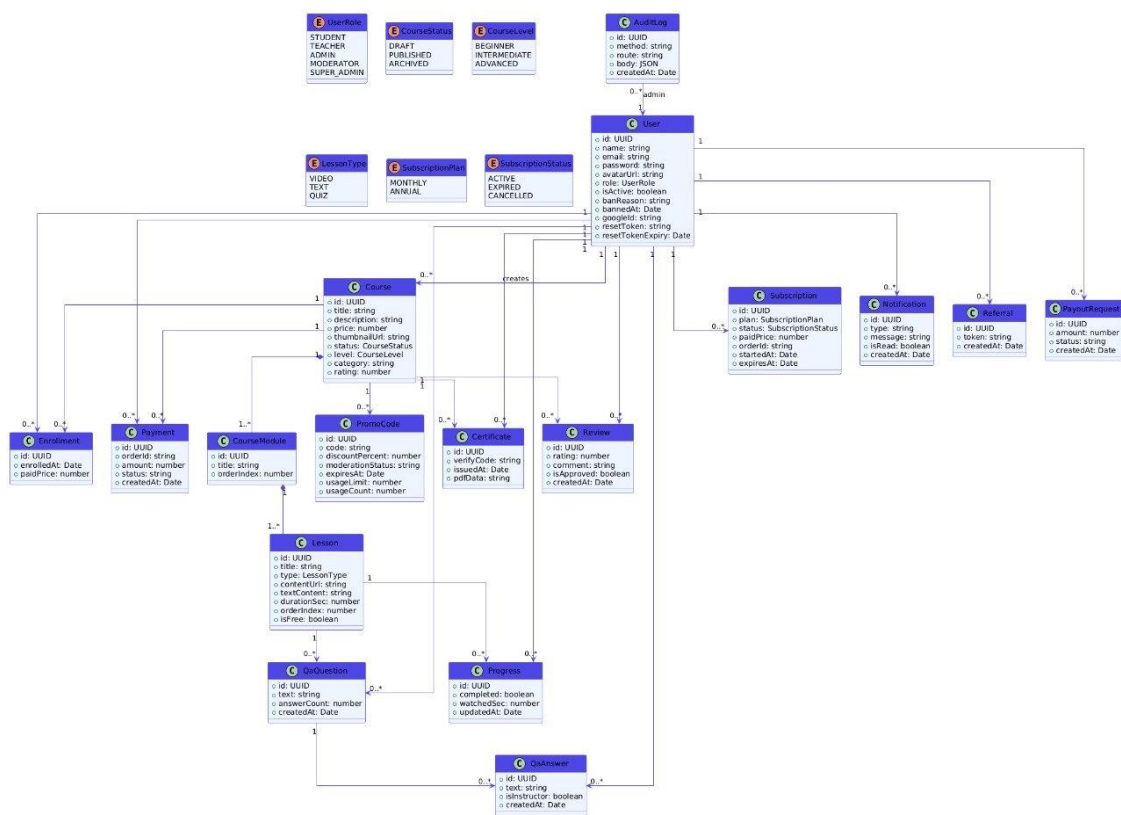


Рис. 2.2 – Діаграма класів системи (UML Class Diagram)

Між сутностями системи встановлено розгалужену мережу зв'язків, що відображає реальну складність освітньої платформи. Зв'язки типу OneToMany

використовуються між User і Enrollment, Course і Enrollment, Course і CourseModule, CourseModule і Lesson, User і Progress, Lesson і Progress. Зв'язки типу ManyToOne відображають зворотні напрямки цих самих відносин. Унікальні складені ключі в сутностях Enrollment та Progress забезпечують цілісність даних і виключають дублювання записів.

## 2.2 Розробка моделей даних та взаємодії з API

Для забезпечення надійної взаємодії клієнтської частини платформи з серверною розроблено систему моделей даних. Серверна частина реалізована на NestJS із використанням TypeORM для роботи з PostgreSQL, а клієнтська - на React із TypeScript. Строга типізація на рівні TypeScript дозволяє виявляти невідповідності типів ще на етапі компіляції, що суттєво підвищує надійність системи і зменшує кількість помилок у виробничому середовищі. Архітектура моделей побудована на принципі розділення DTO для вхідних запитів та entity-класів для представлення даних у базі.

Проектування моделей даних є критично важливим етапом, оскільки правильно спроектована модель забезпечує не лише коректну роботу системи, але й її продуктивність та масштабованість. В умовах освітньої платформи, де дані про курси, уроки, прогрес та платежі тісно пов'язані між собою, якість моделювання безпосередньо впливає на складність запитів до бази даних і швидкість відповіді системи.

Для стандартизації значень атрибутів у системі визначено набір перерахувань (enum). UserRole описує п'ять можливих ролей: STUDENT, TEACHER, ADMIN, MODERATOR, SUPER\_ADMIN. CourseStatus визначає три стани публікації: DRAFT - чернетка, доступна лише автору; PUBLISHED - опублікований курс, видимий у каталозі; ARCHIVED - архівований курс, що не відображається у пошуку. CourseLevel задає три рівні складності: BEGINNER, INTERMEDIATE, ADVANCED, що дозволяє студентам фільтрувати контент відповідно до власного рівня підготовки. LessonType описує три формати навчального матеріалу: VIDEO, TEXT, QUIZ.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 35   |

SubscriptionPlan визначає плани MONTHLY та ANNUAL, а SubscriptionStatus - стани ACTIVE, EXPIRED та CANCELLED.

Використання строго типізованих перерахунків замість рядкових констант забезпечує кілька важливих переваг. По-перше, компілятор TypeScript автоматично виявляє помилки при використанні неприпустимих значень. По-друге, зміна переліку допустимих значень в одному місці автоматично поширюється на всі частини системи. По-третє, IntelliSense в редакторах коду надає підказки з переліком дозволених значень, що прискорює розробку та зменшує ймовірність помилок.

#### Моделі автентифікації та авторизації

Система автентифікації реалізована через JWT із двома видами токенів: access-токеном та refresh-токеном. Access-токен містить у корисному навантаженні ідентифікатор користувача, електронну пошту та роль, має термін дії 15 хвилин і використовується для авторизації кожного запиту до API. Refresh-токен підписується окремим секретним ключем, має термін дії 7 днів і призначений виключно для отримання нового access-токена без повторного введення облікових даних.

Короткий термін дії access-токена є свідомим архітектурним рішенням, що підвищує безпеку системи. Навіть у разі перехоплення токена злоумисник матиме обмежений час для його використання. Refresh-токен зберігається на клієнті та використовується лише для поновлення сесії, що мінімізує ризик його компрометації. Перелік DTO (Data Transfer Objects), що забезпечує повний цикл управління автентифікацією, наведено у таблиці 2.1.

Об'єкт RegisterDto приймає ім'я, електронну пошту, пароль (мінімум 6 символів) та необов'язковий реферальний токен. LoginDto містить лише пару email та пароль. RefreshTokenDto приймає refresh-токен для оновлення сесії. ForgotPasswordDto містить email для ініціації процедури скидання, а ResetPasswordDto - токен підтвердження та новий пароль. На клієнтській стороні AuthUser зберігає ідентифікатор, ім'я, email, роль та URL аватара поточного користувача.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 36   |

## Перелік DTO та інтерфейсів модуля автентифікації

| Об'єкт (DTO / Interface) | Напрямок           | Основні поля                                              |
|--------------------------|--------------------|-----------------------------------------------------------|
| RegisterDto              | Клієнт →<br>Сервер | name, email, password ( $\geq 6$ ), role?, referralToken? |
| LoginDto                 | Клієнт →<br>Сервер | email, password                                           |
| RefreshTokenDto          | Клієнт →<br>Сервер | refreshToken                                              |
| ForgotPasswordDto        | Клієнт →<br>Сервер | email                                                     |
| ResetPasswordDto         | Клієнт →<br>Сервер | token, password ( $\geq 6$ )                              |
| AuthUser                 | Сервер →<br>Клієнт | id, name, email, role, avatarUrl                          |

Клієнтська частина зберігає токени в localStorage браузера. При ініціалізації додатку AuthContext перевіряє наявність access-токена та виконує запит для отримання даних поточного користувача. У разі помилки - наприклад, якщо токен закінчив термін дії - автоматично виконується спроба його поновлення через refresh-токен. Якщо поновлення також завершується помилкою, токени видаляються і користувач вважається неавторизованим. Захист маршрутів реалізовано через компонент ProtectedRoute, що перевіряє роль користувача перед рендерингом сторінки і перенаправляє неавторизованих на сторінку входу.

Для авторизації через Google OAuth реалізована інтеграція з Passport.js на серверній стороні. При натисканні кнопки входу через Google користувач перенаправляється на сторінку авторизації Google, де підтверджує надання доступу до даних профілю. Після успішної авторизації Google повертає профіль користувача на callback-ендпоінт сервера. Система шукає існуючий

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 37   |

обліковий запис за Google ID або email; якщо обліковий запис не знайдено - створює новий з роллю STUDENT. Після цього формується пара JWT-токенів, що передається клієнту через параметри URL перенаправлення.

#### Основні моделі даних платформи

Основні сутності платформи організовані в ієрархічну структуру з чіткими зв'язками між компонентами. Моделі даних спроектовані з урахуванням двох перспектив: серверної - entity-класів TypeORM, що безпосередньо відображаються на таблиці бази даних, та клієнтської - TypeScript-інтерфейсів, що описують форму даних, отриманих від API. Перелік ключових DTO та інтерфейсів наведено у таблиці 2.2.

Таблиця 2.2

#### Основні DTO та інтерфейси платформи

| DTO / Interface      | Рівень        | Ключові поля                                                                 |
|----------------------|---------------|------------------------------------------------------------------------------|
| CreateCourseDto      | Серверний DTO | title, description, price ( $\geq 0$ ), thumbnailUrl?, level?, category?     |
| UpdateCourseDto      | Серверний DTO | Усі поля CreateCourseDto (необов'язкові) + status?                           |
| CourseFilterDto      | Серверний DTO | search?, category?, level?, page?, limit?, minPrice?, maxPrice?, minRating?  |
| CreateModuleDto      | Серверний DTO | title, orderIndex?                                                           |
| CreateLessonDto      | Серверний DTO | title, type, contentUrl?, textContent?, durationSec?, orderIndex?, isFree?   |
| UpdateProgressDto    | Серверний DTO | lessonId (UUID), completed (bool), watchedSec ( $\geq 0$ )                   |
| Course (interface)   | Клієнтський   | id, title, description, price, level, rating, author, modules[], isEnrolled? |
| Progress (interface) | Клієнтський   | percent, completedCount, totalCount, completedLessonIds[]                    |

CreateCourseDto визначає поля для створення нового курсу: назву, опис, ціну з мінімальним значенням 0, необов'язкові URL обкладинки, рівень складності та категорію. UpdateCourseDto включає всі ті самі поля як необов'язкові, а також поле зміни статусу. CourseFilterDto описує параметри фільтрації та пагінації для каталогу. CreateModuleDto і CreateLessonDto визначають необхідні поля для створення відповідних структурних елементів курсу. UpdateProgressDto приймає ідентифікатор уроку, прапор завершення та кількість переглянутих секунд.

На клієнтській стороні інтерфейс Course агрегує всі дані курсу, включаючи вкладену структуру модулів та уроків, а також поле isEnrolled, що показує, чи записаний поточний студент. Інтерфейс Progress описує стан проходження курсу: відсоток завершення, кількість завершених уроків, загальну кількість та ідентифікатори завершених уроків. Останнє поле є ключовим для відображення стану окремих уроків у навчальному плані без додаткових запитів до сервера.

Важливим принципом при проєктуванні моделей даних є розмежування між тим, які дані зберігаються на сервері, та тим, які надсилаються клієнту. Наприклад, хеш пароля та токен скидання пароля ніколи не включаються у відповіді API - для цього реалізована спеціальна функція "безпечного" перетворення об'єкта користувача перед його відправкою. Аналогічно, поле contentUrl уроку, що містить посилання на відеоматеріал, передається клієнту лише для записаних студентів, власника курсу та адміністраторів.

REST API серверної частини задокументовано через Swagger (OpenAPI). Кожен контролер NestJS використовує спеціальні декоратори для автоматичної генерації документації, що завжди залишається актуальною і відповідає реальному стану коду. Для вхідних DTO застосовується декларативна валідація через class-validator, а глобальний ValidationPipe автоматично відкидає невідомі поля та виконує приведення типів. Основні групи ендпоінтів наведено у таблиці 2.3.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 39   |

## Перелік ендпоінтів REST API платформи

| Модуль            | Ендпоінти                                                                                                    | Призначення                                                   |
|-------------------|--------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| Auth              | POST /register, /login, /refresh; GET /me, /google, /google/callback; POST /forgot-password, /reset-password | Реєстрація, вхід, OAuth, оновлення токена, скидання пароля    |
| Courses           | GET /courses, /:id, /:id/progress; POST /, /:id/enroll; PATCH /:id, /progress/save                           | Каталог, перегляд, запис, прогрес, CRUD курсу                 |
| Modules / Lessons | POST /modules/:moduleId/lessons; PATCH /modules/:id, /lessons/:id; DELETE /modules/:id, /lessons/:id         | Управління структурою курсу, каскадне видалення               |
| Payments          | POST /payments/create/:courseId, /subscribe, /callback; GET /status/:courseId                                | Ініціація оплати, обробка webhook WayForPay, підписки         |
| Certificates      | POST /issue/:courseId; GET /my, /download/:code, /verify/:code                                               | Видача, перелік, завантаження та верифікація PDF-сертифікатів |
| Storage           | POST /upload/video, /upload/image; GET /upload/video-url/:key                                                | Завантаження медіафайлів, отримання підписаних URL            |
| Analytics         | GET /analytics/student, /teacher, /courses/:id                                                               | Статистика студента, викладача, деталі по курсу               |
| Admin             | GET /admin/stats, /users, /courses; PATCH /users/:id; POST /users/:id/ban, /unban; PATCH /courses/:id/status | Управління користувачами, статус курсів, адмін-аналітика      |
| QA / Reviews      | GET /qa/lesson/:id; POST /questions, /answers; GET /reviews/:courseId; POST /reviews/:courseId               | Q&A до уроків, відгуки та рейтинги курсів                     |

Модуль Auth забезпечує реєстрацію, вхід, оновлення токена, авторизацію через Google, скидання пароля та отримання даних поточного користувача. Модуль Courses надає ендпоінти для отримання каталогу з фільтрацією, перегляду одного курсу, отримання прогресу, запису на курс та збереження прогресу уроку. Modules і Lessons надають CRUD-операції для структурних елементів курсу з каскадним видаленням і автоматичним встановленням порядкового номера.

Модуль Payments обробляє ініціацію оплати та отримання callback від WayForPay. Certificates забезпечує видачу, перелік, верифікацію та завантаження PDF-сертифікатів. Subscriptions реалізує оформлення підписки, перегляд поточного стану та скасування. Модуль Storage надає ендпоінти для завантаження відео та зображень, а також для отримання підписаних URL для приватних відеоматеріалів. QA забезпечує постановку запитань і відповідей з прив'язкою до уроку.

Окрему групу утворюють аналітичні ендпоінти. Для студента доступна статистика навчальної активності: загальний час перегляду, поточна серія активних днів, розподіл активності за годинами та днями тижня. Для викладача - зведені дані по всіх курсах: кількість студентів, дохід, кількість виданих сертифікатів. Адміністративна аналітика включає статистику платформи з динамікою реєстрацій, доходу та рейтингом найпопулярніших курсів.

На серверній стороні валідація вхідних даних реалізована через декларативні анотації у DTO-класах. Цей підхід відділяє логіку валідації від бізнес-логіки та забезпечує однорідне оброблення помилок. Глобальний ValidationPipe з увімкненим whitelist:true автоматично видаляє будь-які поля, що не оголошені у DTO, навіть якщо вони присутні у тілі запиту. Це захищає від атак масового присвоєння (mass assignment), коли злоумисник намагається встановити значення полів, що не призначені для редагування. Параметр transform:true дозволяє автоматично перетворювати рядкові значення з query string на числа або булеві типи відповідно до оголошення у DTO.

Для поля пароля при реєстрації встановлено мінімальну довжину 6 символів. Поля email перевіряються на відповідність стандартному формату. Числові поля ціни мають мінімальне значення 0 для запобігання від'ємних значень. Поля перерахувань валідуються на відповідність одному з дозволених значень. Необов'язкові поля позначаються відповідним декоратором і не викликають помилок валідації у разі їхньої відсутності.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 41   |

Крім серверної валідації, клієнтська частина також виконує перевірку форм перед відправкою запиту. Це зменшує кількість зайвих мережових запитів і надає користувачу миттєвий зворотній зв'язок без очікування відповіді сервера. Поєднання клієнтської та серверної валідації забезпечує захист як для зручності використання, так і для безпеки.

Глобальний стан React-додатку управляється через Context API - вбудований механізм React для передачі даних між компонентами без прокидання через prop. Передбачено чотири провайдери: AuthContext управляє станом автентифікації і надає методи для входу, реєстрації, виходу та авторизації через Google. ThemeContext зберігає поточну тему інтерфейсу - темну або світлу - і синхронізує її з localStorage для збереження між сесіями. WishlistContext управляє списком бажаних курсів і синхронізує його стан із сервером при кожній зміні. NotificationContext зберігає системні сповіщення з лічильником непрочитаних для відображення в інтерфейсі навігації.

Для взаємодії з API реалізована централізована утилітна функція apiFetch, що є обгорткою над нативним Fetch API. Вона автоматично додає JWT-заголовок авторизації з токена, збереженого в localStorage, встановлює заголовок Content-Type та обробляє помилки HTTP. У разі помилкової відповіді сервера функція намагається розпарсити JSON-тіло відповіді для отримання повідомлення про помилку і перетворює його на об'єкт Error. Це забезпечує однорідне оброблення помилок у всіх частинах клієнтської програми.

Для отримання даних у компонентах використовуються кастомні React-хуки. Хук useCourses приймає об'єкт параметрів фільтрації та підписки на зміну каталогу курсів. Хук useCourse завантажує деталі одного курсу за ідентифікатором. Хук useCourseProgress відстежує прогрес поточного студента для конкретного курсу і надає функцію примусового оновлення. Хук useMyEnrollmentsProgress завантажує прогрес по всіх записаних курсах для відображення на сторінці каталогу. Хуки useCourseActions надають методи для запису, оновлення прогресу та отримання сертифіката. Така структура

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 42   |

дозволяє повторно використовувати логіку взаємодії з API і забезпечує чистоту компонентів.

## 2.3 Проєктування та реалізація алгоритмів роботи системи

Ефективне функціонування вебплатформи онлайн-навчання забезпечується комплексом взаємопов'язаних алгоритмів, що реалізують ключові бізнес-процеси системи. Кожен алгоритм спроектований з урахуванням вимог до безпеки, продуктивності та стійкості до помилок. У межах даного підрозділу описано алгоритми автентифікації та авторизації, управління навчальним процесом, обробки платіжних транзакцій, автоматичного формування сертифікатів та модерації контенту з аудитом.

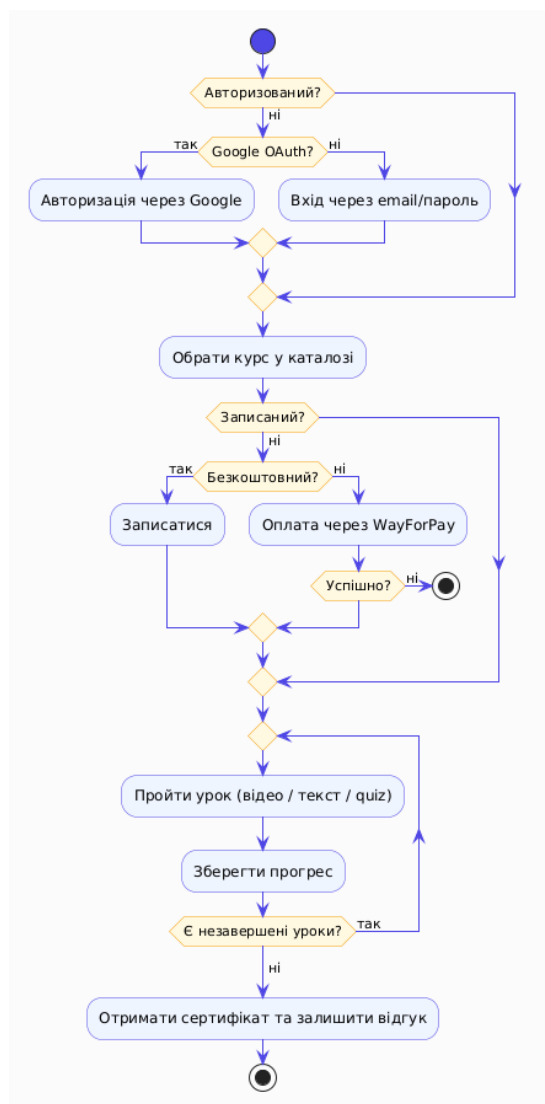


Рис. 2.3 – Загальний алгоритм роботи з платформою

Процес автентифікації є найбільш критичним з точки зору безпеки і охоплює три сценарії: вхід через форму з email і паролем, OAuth-авторизацію через Google та автоматичне поновлення токенів. Кожен сценарій реалізований із власними механізмами захисту та обробки помилок.

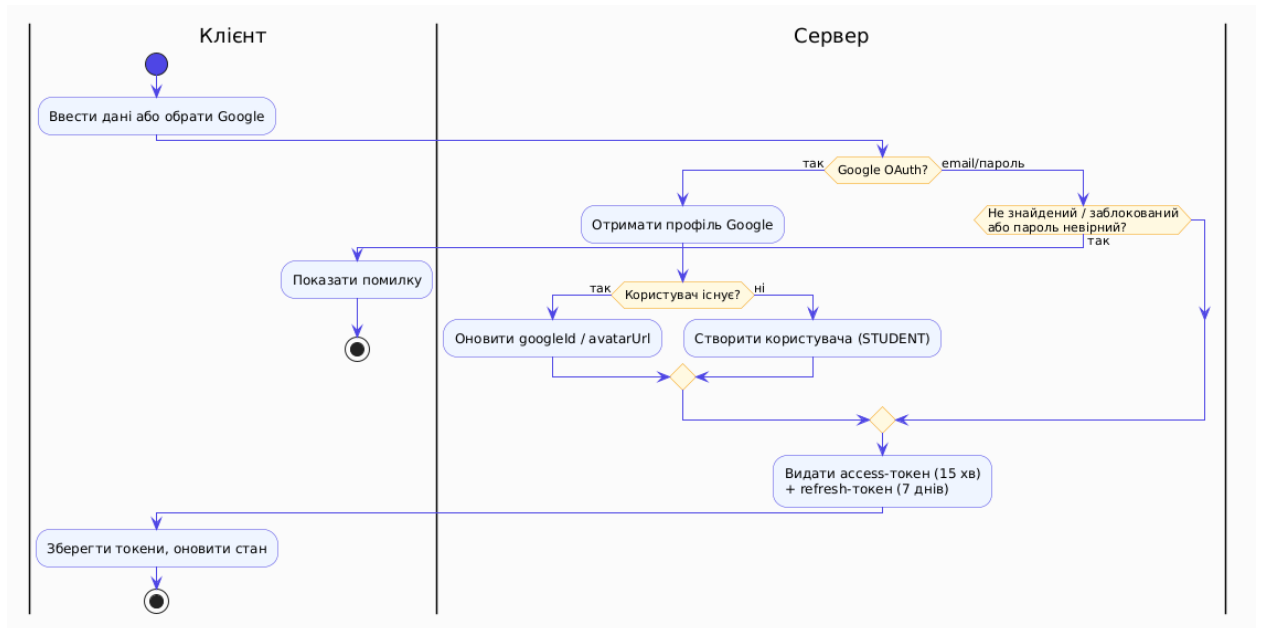


Рис. 2.4 – Діаграма активності процесу автентифікації

Алгоритм входу через форму виконується у кілька послідовних кроків. Спочатку система шукає користувача в базі даних за адресою електронної пошти. Якщо обліковий запис не знайдено, повертається помилка автентифікації з узагальненим повідомленням - таке формулювання не дозволяє зловмиснику з'ясувати, чи зареєстрований певний email на платформі. Наступним кроком перевіряється прапор активності облікового запису: заблоковані користувачі отримують повідомлення із зазначеною причиною блокування. Після успішної перевірки статусу виконується порівняння введеного пароля з хешом, збереженим у базі даних, через бібліотеку bcrypt. Бібліотека bcrypt використовує алгоритм адаптивного хешування з salt, що робить брутфорс-атаки практично неможливими навіть при отриманні зловмисником доступу до бази даних. У разі успішного співставлення паролів формується пара access та refresh токенів і повертається відповідь із даними користувача.

Алгоритм Google OAuth реалізований через NestJS Passport-стратегію, що обробляє весь цикл взаємодії з Google API. Після отримання профілю від провайдера сервер виконує пошук існуючого облікового запису спочатку за Google ID, потім за email. Якщо знайдено обліковий запис лише за email - він оновлюється: до нього прив'язується Google ID і за необхідності встановлюється аватар. Якщо обліковий запис не існує - створюється новий з роллю STUDENT та отриманими від Google даними профілю. Після успішного знаходження або створення облікового запису виконується перевірка його активності та формується пара JWT-токенів. Для нових користувачів додатково надсилається сповіщення адміністраторам про нову реєстрацію.

Алгоритм скидання пароля реалізований через двоетапну процедуру. На першому етапі система знаходить обліковий запис за email (якщо email не існує в системі - відповідь не відрізняється від успішної, щоб унеможливити перебір адрес), генерує криптографічно стійкий токен, обчислює його SHA-256-хеш та зберігає у базі даних разом із часом закінчення терміну дії (1 година). Користувачу надсилається email з посиланням, що містить оригінальний токен. На другому етапі система знаходить обліковий запис за хешем токена (зберігання хешу, а не самого токена, захищає навіть у разі витоку бази даних), перевіряє термін дії і оновлює пароль. Після успішного скидання токен негайно видаляється з бази даних, щоб виключити повторне використання.

Захист маршрутів API реалізований через глобальний JwtAuthGuard, який перехоплює кожен запит до захищених ендпоінтів, верифікує підпис JWT та заповнює об'єкт поточного користувача в контексті запиту. RolesGuard перевіряє відповідність ролі поточного користувача вимогам, оголошеним через декоратор на контролері або методі. Поєднання цих двох механізмів забезпечує гнучку рольову модель доступу.

Центральним алгоритмом платформи є логіка запису на курс, відстеження прогресу та визначення моменту завершення навчання. Цей

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 45   |

алгоритм безпосередньо впливає на навчальний досвід студента і повинен забезпечувати коректне відстеження стану навіть у разі паралельних запитів.

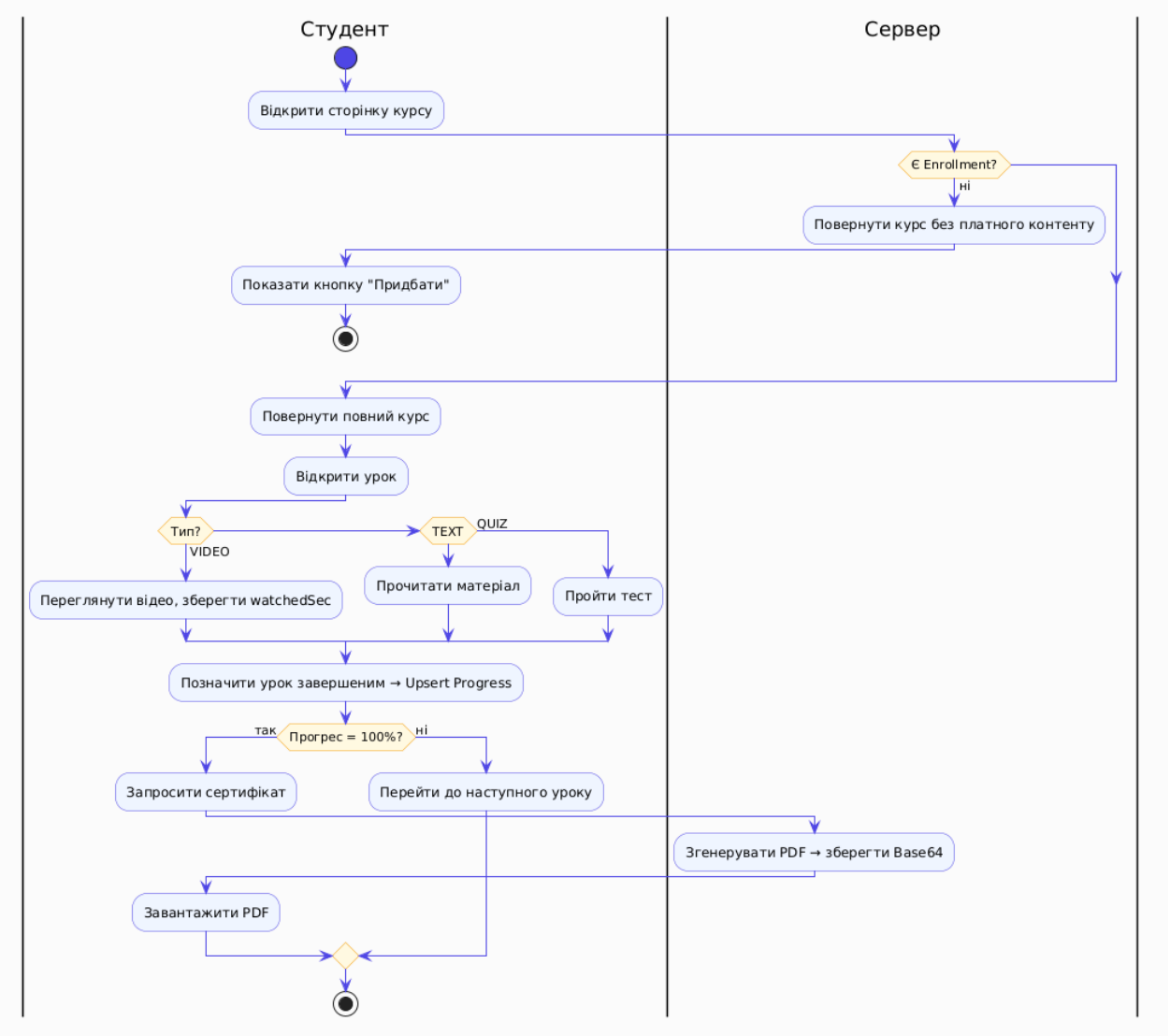


Рис. 2.5 – Діаграма активності процесу проходження курсу

При отриманні запиту на сторінку курсу система завантажує повну структуру курсу разом з ієрархією модулів та уроків і перевіряє наявність запису Enrollment для поточного користувача. Логіка доступу до контенту реалізована безпосередньо на сервері: якщо користувач не записаний на курс і не є власником або адміністратором, поле contentUrl усіх уроків із прапором isFree=false замінюється значенням null. Таким чином, навіть якщо клієнтська частина була б модифікована зловмисником, медіаконтент залишатиметься недоступним без запису.

Запис на безкоштовний курс виконується безпосередньо через ендпоінт enroll. Алгоритм перевіряє існування курсу та відсутність дублікату запису для

даної пари користувач-курс. При виявленні спроби повторного запису повертається помилка ConflictException. Після успішного запису асинхронно надсилаються два сповіщення: студенту про успішний запис і викладачу про нового студента. Використання асинхронної відправки через fireAndForget запобігає затриманню відповіді API через можливу повільну роботу сервісу сповіщень.

Алгоритм збереження прогресу реалізований через операцію upsert: якщо запис Progress для даної пари userId та lessonId вже існує - він оновлюється, якщо ні - створюється новий. Перед збереженням система перевіряє, чи має користувач право оновлювати прогрес: власники курсу та адміністратори можуть оновлювати без запису, тоді як студенти повинні мати активний Enrollment. Кількість переглянутих секунд оновлюється за принципом максимуму - якщо нове значення менше від збереженого, воно ігнорується. Це запобігає випадковому зменшенню прогресу через помилки у відстеженні на клієнті.

Відсоток завершення курсу обчислюється динамічно при кожному запиті. Алгоритм завантажує повну структуру курсу, збирає ідентифікатори всіх уроків, підраховує кількість записів Progress зі статусом completed=true для поточного користувача та обчислює відношення завершених до загальних. Результат округлюється до цілого відсотка. Разом із відсотком повертається список ідентифікаторів завершених уроків, що дозволяє клієнту відображати стан кожного уроку без додаткових запитів.

Для відеоуроків реалізовано механізм автоматичного зарахування: клієнтська частина кожні 30 секунд надсилає на сервер поточну позицію відтворення, а також позначає урок як завершений, коли відтворено більше 80% тривалості. Цей поріг є компромісом між суворою вимогою повного перегляду та врахуванням випадків, коли студент може вже знати матеріал і пропустити ознайомчу або підсумкову частину. Для YouTube-відео відстеження через HTML5 API неможливе через обмеження браузера, тому для них надається кнопка ручного підтвердження завершення.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 47   |

Процес придбання курсу реалізований через двоетапний алгоритм із залученням платіжного шлюзу WayForPay. Вибір WayForPay зумовлений підтримкою платежів у гривнях та наявністю зручного API для серверної інтеграції.

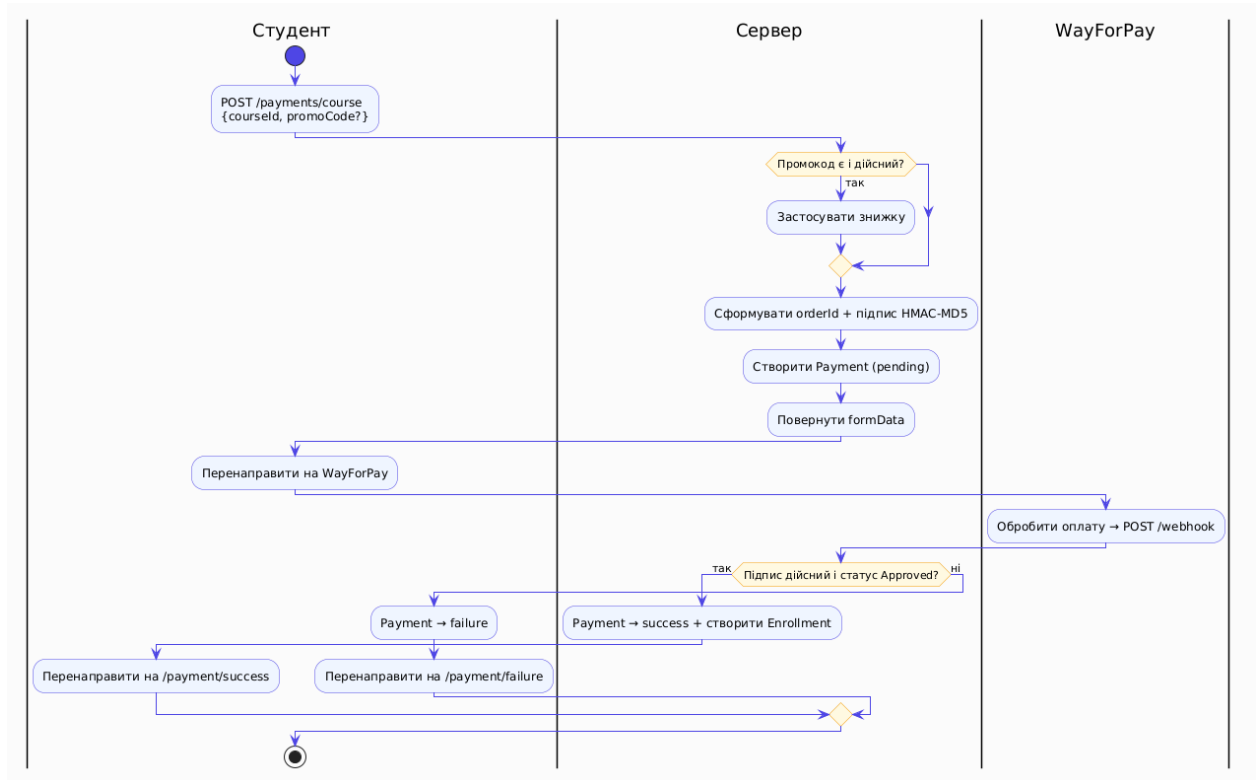


Рис. 2.6 – Діаграма активності процесу оплати курсу

На першому етапі сервер формує дані платіжної форми. Генерується унікальний ідентифікатор замовлення у форматі, що кодує ідентифікатори курсу та користувача разом з випадковим рядком для унікальності. Обчислюється цифровий підпис HMAC-MD5[8] з обов'язковими параметрами: ідентифікатор мерчанта, домен, ідентифікатор замовлення, дата, сума, валюта, назва продукту, кількість та ціна. Клієнт отримує об'єкт formData та URL платіжної сторінки. Для застосування промокоду сума зменшується на відповідний відсоток перед формуванням підпису.

Клієнтська частина обробляє отримані дані шляхом динамічного створення HTML-форми з прихованими полями та автоматичного її відправлення. Це перенаправляє браузер на захищену сторінку WayForPay, де студент вводить платіжні дані. Такий підхід відповідає вимогам безпеки PCI

DSS: платіжні дані картки ніколи не проходять через сервери платформи і не можуть бути перехоплені.

На другому етапі, після завершення транзакції, WayForPay надсилає асинхронне підтвердження (callback) на серверний ендпоінт POST /payments/webhook. Алгоритм верифікує підпис отриманого повідомлення: обчислює очікуваний HMAC-MD5 на основі отриманих параметрів та порівнює з підписом у повідомленні. Якщо підписи не збігаються - запит відхиляється як підроблений. Після успішної верифікації алгоритм аналізує статус транзакції: при статусі Approved декодує ідентифікатори курсу та користувача з ідентифікатора замовлення, створює записи Enrollment і Payment зі статусом success. При будь-якому іншому статусі створюється лише запис Payment зі статусом failure. Незалежно від результату, сервер повертає WayForPay стандартну відповідь-підтвердження, підписану HMAC-MD5. Відсутність цієї відповіді призведе до повторних спроб WayForPay надіслати callback.

Аналогічний алгоритм реалізовано для підписки. Ідентифікатор замовлення кодує тип операції, план підписки та ідентифікатор користувача. Після успішного підтвердження автоматично встановлюється дата закінчення підписки: плюс 30 днів для місячного плану або плюс 365 днів для річного. Наявна підписка перевіряється при кожному запиті до відповідних захищених ендпоінтів.

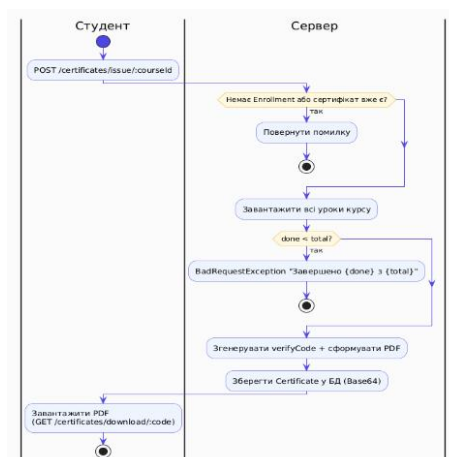


Рис. 2.7 – Діаграма активності процесу формування сертифіката

При запиті на видачу сертифіката CertificateService виконує комплексну перевірку відповідності вимогам. Перший крок - перевірка наявності активного запису Enrollment, що підтверджує факт запису студента на курс. Другий крок - перевірка відсутності вже виданого сертифіката для уникнення дублікатів. Третій крок - завантаження повної структури курсу з усіма модулями та уроками для підрахунку загальної кількості навчальних одиниць.

Після отримання структури курсу алгоритм збирає ідентифікатори всіх уроків і підраховує кількість записів Progress зі статусом завершення для поточного студента. Якщо кількість завершених уроків менша від загальної - повертається помилка з точним повідомленням про кількість пройдених та необхідних уроків. Вимога 100-відсоткового завершення є свідомим архітектурним рішенням, що підвищує цінність сертифікату: він підтверджує повне проходження програми, а не лише часткове знайомство з матеріалом.

При підтвердженні повного прогресу генерується 12-байтний криптографічно стійкий код верифікації у шістнадцятковому форматі. Цей код є унікальним ідентифікатором сертифіката, що дозволяє будь-кому перевірити його автентичність через публічний ендпоінт без необхідності автентифікації.

Формування PDF-документа здійснюється через бібліотеку pdfkit у форматі A4-landscape. Документ має структуровану композицію: золота подвійна рамка, заголовок платформи, підзаголовок про завершення курсу, горизонтальний роздільник, повне ім'я студента, назва курсу з виділенням кольором акценту, дата видачі, ім'я викладача та код верифікації. Для відображення кирилиці завантажуються власні TTF-файли шрифту Roboto, оскільки стандартні шрифти pdfkit не підтримують кирилицю. Готовий PDF-документ конвертується в рядок Base64 та зберігається безпосередньо в базі даних. Такий підхід забезпечує незалежність від зовнішніх файлових сховищ і гарантує доступність сертифіката в будь-який момент.

Після успішної видачі сертифіката асинхронно надсилається сповіщення студенту з кодом верифікації. Студент може завантажити PDF через

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 50   |

захищений ендпоінт, що повертає бінарні дані з відповідними заголовками для автоматичного відкриття діалогу збереження у браузері.

Система модерації контенту забезпечує якість матеріалів на платформі через дворівневий контроль. Перший рівень - автоматичний: при створенні або оновленні курсу надсилається сповіщення адміністраторам про необхідність перевірки. Новий курс автоматично отримує статус DRAFT і не відображається у публічному каталозі до прийняття рішення модератором. Другий рівень - ручний: адміністратор або модератор переглядає курс та змінює його статус на PUBLISHED або залишає у статусі DRAFT, повертаючи на доопрацювання.

При зміні статусу курсу на PUBLISHED викладач автоматично отримує сповіщення, а студентам, що підписані на нові курси, надсилається інформація про появу нового контенту. При архівуванні курсу він зникає з каталогу, але зберігається для вже записаних студентів. Система відгуків також проходить модерацію: новий відгук зберігається з прапором `isApproved=false` і не відображається публічно до його схвалення адміністратором.

Усі адміністративні дії фіксуються через глобальний NestJS-інтерсептор, що перехоплює POST, PATCH та DELETE-запити до маршрутів адмін-панелі. Для кожного такого запиту автоматично зберігається запис аудиту: HTTP-метод, маршрут, ідентифікатор адміністратора та тіло запиту. Це забезпечує повну прозорість адміністративних дій і дозволяє відстежити, хто, коли і які зміни вніс у систему. Журнал аудиту доступний через окремий ендпоінт тільки для адміністраторів і супер-адміністраторів.

Захист від перевантаження реалізовано через глобальний ThrottlerGuard: максимум 100 запитів за 60 секунд для будь-якого ендпоінту. Це запобігає як DDoS-атакам, так і зловживанню брутфорс-атаками на ендпоінт автентифікації. Ліміт є достатньо великим для нормального використання і достатньо малим для обмеження автоматизованих атак.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 51   |

## 2.4 Реалізація функціоналу системи вебплатформи для онлайн-навчання

У цьому підрозділі розглянуто основні аспекти практичної реалізації функціоналу вебплатформи онлайн-навчання. Реалізація охоплює дві взаємопов'язані частини: серверну на NestJS та клієнтську на React і TypeScript. Основна увага приділяється архітектурним рішенням, що забезпечують масштабованість, безпеку та якість користувацького досвіду.

Серверна частина реалізована як REST API на NestJS із модульною архітектурою, де кожна предметна область виокремлена в окремий NestJS-модуль. Система включає наступні модулі: AuthModule відповідає за всі аспекти автентифікації та авторизації; CoursesModule управляє навчальним контентом, записом та прогресом; CertificateModule реалізує видачу та верифікацію сертифікатів; PaymentModule обробляє фінансові транзакції через WayForPay; SubscriptionModule управляє підписками; QaModule та ReviewModule реалізують систему запитань-відповідей і відгуків; NotificationModule забезпечує доставку сповіщень; PromoCodeModule управляє промо-акціями; WishlistModule реалізує список бажаних; ReferralModule відстежує реферальні програми; PayoutModule обробляє запити на виплату; InstructorModule надає дані профілю викладача; AnalyticsModule збирає та агрегує аналітичні дані; AdminModule реалізує адміністративний функціонал; AuditModule веде журнал дій; StorageModule відповідає за роботу з медіафайлами.

Така модульна структура забезпечує ізоляцію відповідальності, незалежне тестування компонентів та можливість масштабування окремих частин системи. Кожен модуль містить контролер, сервіс та необхідні entity-класи. Контролери відповідають за маршрутизацію запитів та валідацію вхідних даних, сервіси - за реалізацію бізнес-логіки, entity-класи - за відображення на таблиці бази даних.

Взаємодія з базою даних PostgreSQL реалізована через TypeORM. Схема бази даних визначається виключно через entity-класи з декораторами

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 52   |

TypeORM, а всі зміни схеми управляються через систему міграцій. Платформа включає 15 послідовних міграцій, що охоплюють усю еволюцію схеми: від початкової структури з основними таблицями до додавання сповіщень, промокодів, вішлісту, блокування, аудиту, Google OAuth, реферальної системи, виплат, системи Q&A, підписок та скидання пароля. Використання міграцій замість автоматичного synchronize гарантує контрольоване та передбачуване оновлення схеми у виробничому середовищі.

StorageService розрізняє два типи доступу залежно від характеру контенту: відеоматеріали уроків зберігаються як приватні об'єкти і видаються через підписані URL з терміном дії 1 година, тоді як зображення обкладинок курсів та аватарів користувачів зберігаються з публічним доступом. Контролер завантаження приймає файли через multipart/form-data із перевіркою типу MIME та розміру: відео до 500 МБ, зображення до 5 МБ. Таке обмеження захищає від зловживання сховищем і забезпечує прийнятний час завантаження для кінцевих користувачів.

Для первинного наповнення бази даних реалізовано seed-скрипт, що створює тестових користувачів різних ролей, набір курсів різного типу та кілька записів прогресу. Це дозволяє швидко розгорнути повністю функціональний демонстраційний стенд без ручного введення даних.

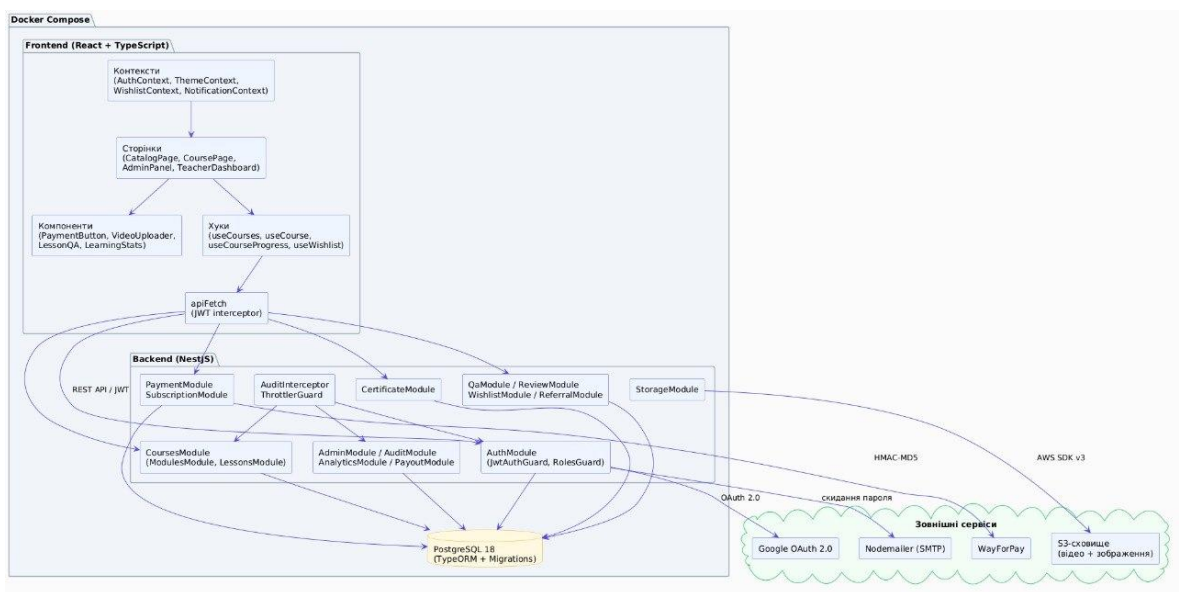


Рис. 2.8 – Діаграма системної архітектури додатку

Клієнтська частина побудована на React з TypeScript як SPA із використанням Create React App. Маршрутизація реалізована через React Router DOM з декларативним оголошенням маршрутів. Архітектуру клієнта організовано за чотирма шарами з чіткою відповідальністю кожного.

Шар компонентів сторінок (pages/) відповідає за композицію та відображення даних для конкретних маршрутів. Кожна сторінка є контейнером, що використовує хуки для отримання даних і передає їх у компоненти інтерфейсу. Шар компонентів інтерфейсу (components/) містить повторно використовувані елементи: Navbar забезпечує навігацію та відображення стану автентифікації; Footer містить посилання та інформацію про платформу; ProtectedRoute перевіряє права доступу; Skeleton відображає анімовані заглушки під час завантаження; Toast показує тимчасові сповіщення; PaymentButton реалізує інтеграцію з WayForPay; WishlistButton керує додаванням до списку бажаних; NotificationBell відображає сповіщення; VideoUploader забезпечує завантаження відео з прогрес-індикатором; LessonQA реалізує систему питань і відповідей у контексті уроку; LearningStats показує зведену статистику навчання.

Шар контекстів (context/) реалізує глобальний стан через React Context API. Кожен провайдер відповідає за свою область: автентифікація, тема, вішліст та сповіщення. Компоненти підписуються лише на потрібні контексти, що мінімізує зайвий рендеринг. Шар хуків (hooks/) містить кастомні React-хуки для взаємодії з API: useCourses, useCourse, useCourseProgress, useMyEnrollmentsProgress, useCourseActions та useWishlist. Ці хуки інкапсулюють логіку мережевих запитів і управління локальним станом, надаючи компонентам чистий інтерфейс для роботи з даними.

CatalogPage реалізує багаторівневу фільтрацію курсів через єдиний зручний інтерфейс. Стан фільтрів підтримується в локальному стані компонента та синхронізується з URL-параметрами, що дозволяє зберігати та ділитися посиланнями на відфільтровані результати. Хук useCourses формує

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 54   |

query string із поточних параметрів фільтрації та відправляє запит до API при зміні будь-якого фільтра або сторінки пагінації.

Список категорій генерується динамічно на основі категорій наявних курсів, що усуває потребу в окремому ендпоінті для їхнього отримання. Фільтрація за рівнем відображається у вигляді радіо-кнопок із чіткими підписами, фільтрація за ціною - через повзунок діапазону, фільтрація за рейтингом - через інтерактивні зірки. Кнопка скидання фільтрів відображається лише тоді, коли застосований принаймні один фільтр, відмінний від значень за замовчуванням.

Результати пошуку підтримують два режими відображення: сітку карток та список. У режимі сітки кожна картка містить зображення або автоматично згенерований градієнтний фон на основі першої літери назви, рівень складності у вигляді бейджа, значок "FREE" для безкоштовних курсів, назву, короткий опис, рейтинг у вигляді зірок, ім'я автора та ціну. Для записаних студентів відображається прогрес-бар з відсотком завершення. У режимі списку відображається компактніший варіант з thumbnail і основними метаданими.

Компонент Skeleton відображає анімовані заглушки під час завантаження - за структурою вони ідентичні реальним карткам, що запобігає різкому "стрибку" інтерфейсу при отриманні даних. Пагінація реалізована з відображенням "..." для великого діапазону сторінок - відображаються перша, остання та три навколо поточної.

CoursePage є центральним елементом взаємодії студента з платформою та реалізує найбільш складний сценарій взаємодії. Сторінка має триколонкову структуру: хедер з загальною інформацією про курс і карткою дії, ліва колонка з навчальним планом та права колонка з плеєром активного уроку.

Хедер містить хлібні крихти для навігації, заголовки і опис курсу, метадані (автор, кількість уроків, загальна тривалість, рівень складності) та картку з ціною, прогресом і кнопкою дії. Кнопка дії є динамічною: для власника курсу відображається статична мітка, для адміністратора - режим

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 55   |

перегляду, для не записаних студентів при безкоштовному курсі - кнопка запису, при платному - компонент PaymentButton. Для записаних студентів при незавершеному курсі - кнопка продовження, при 100-відсотковому завершенні - кнопка отримання сертифіката. Прогрес-бар у картці відображається лише для записаних студентів і показує відсоток завершення разом із кількістю пройдених уроків.

Ліва бічна панель відображає навчальний план у вигляді ієрархії модулів і уроків. Модулі можна розгортати та згорнути кліком по заголовку. Кожен урок відображає іконку типу (відео, текст або quiz), назву та тривалість. Заблоковані уроки (платні для не записаних студентів) відображаються з напівпрозорістю і не реагують на клік. Завершені уроки позначаються зеленою галочкою. Активний урок виділяється контрастним фоном. При зміні активного уроку правий плеєр оновлюється відповідним контентом.

Компонент LessonPlayer обробляє три типи контенту. Для відеоуроків перевіряється формат посилання: YouTube-відео вбудовуються через iframe для максимальної сумісності. Завершення фіксується автоматично при перегляді 80% або через кнопку ручного підтвердження для YouTube. Для текстових уроків відображається HTML-контент (конвертований з Markdown на сервері) з кнопкою підтвердження. Для quiz-уроків компонент QuizPlayer відображає запитання з варіантами відповідей, перевіряє результат та відображає пояснення для кожного запитання після відповіді.

Після кожного збереження прогресу система перевіряє, чи досяг студент 100-відсоткового завершення курсу. Якщо так - автоматично ініціюється запит на видачу сертифіката. Для запобігання дублювання запитів використовується ref-прапор, що блокує повторне виконання до завершення першого. При успішному отриманні сертифіката відображається модальне вікно з вітанням, кнопкою завантаження PDF та кодом верифікації, а також формою для залишення відгуку про курс.

Компонент PaymentButton реалізує повний цикл взаємодії з платіжним API. При натисканні кнопки відправляється запит до серверного ендпоінту з

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 56   |

ідентифікатором курсу та промокодом, якщо він введений у відповідне поле. Сервер повертає дані форми та URL платіжної сторінки WayForPay. Компонент динамічно створює HTML-форму, встановлює всі поля через приховані inputs та ініціює автоматичне відправлення, перенаправляючи браузер на захищену сторінку оплати.

Після повернення з платіжної сторінки PaymentResultPage обробляє параметри URL: при статусі success відображається підтвердження з посиланням на курс, при failure - повідомлення про помилку з пропозицією повторити спробу. Сторінка не залежить від webhook-підтвердження, оскільки Enrollment може ще не бути створено у момент повернення - callback від WayForPay може прийти з невеликою затримкою.

SubscriptionPage дозволяє вибрати план підписки та відображає порівняння переваг. При виборі плану аналогічним чином формується форма WayForPay і виконується перенаправлення. Активна підписка відображається на сторінці профілю з датою закінчення та кнопкою скасування. Скасування підписки не дає повернення коштів і лише змінює статус на CANCELLED без негайного відключення доступу - студент зберігає доступ до кінця оплаченого терміну.

MyCertificatesPage завантажує та відображає список усіх отриманих студентом сертифікатів. Кожен сертифікат у списку показує назву курсу, ім'я викладача, дату видачі та код верифікації. Для кожного сертифіката доступна кнопка завантаження PDF, що ініціює запит до ендпоінту GET /certificates/download/:code. Сервер повертає бінарні дані з заголовком Content-Disposition: attachment, що ініціює діалог збереження файлу у браузері.

VerifyCertPage реалізує публічну перевірку сертифіката, доступну без автентифікації. Користувач вводить код верифікації у форму, і система відображає деталі: ім'я студента, назву курсу та дату видачі. Ця сторінка може бути використана роботодавцями або освітніми установами для перевірки автентичності сертифіката без необхідності звертатися до платформи.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 57   |

TeacherDashboard є центральним робочим простором для викладача і відображає зведену статистику: загальну кількість студентів, дохід, кількість виданих сертифікатів та середній рейтинг по всіх курсах. Нижче відображається список курсів зі статусами, кількістю записів та швидкими діями - переходом до редагування та аналітики.

CourseCreatePage та CourseEditPage реалізують форми для роботи з курсом. Базова інформація включає назву, опис, ціну, рівень та категорію. Для завантаження обкладинки інтегровано VideoUploader з підтримкою попереднього перегляду. Після збереження основних даних відкривається розширений редактор для управління структурою курсу: додавання, редагування та видалення модулів і уроків з drag-and-drop впорядкуванням. Для кожного уроку можна вказати тип, завантажити відео або текстовий вміст, налаштувати прапор безкоштовного доступу.

CourseAnalyticsPage завантажує детальну аналітику по конкретному курсу: динаміку записів по днях за останні 30 днів у вигляді графіка, середній відсоток прогресу студентів, список найпопулярніших уроків за кількістю переглядів та середнім часом перегляду. Ці дані допомагають викладачу зрозуміти, які частини курсу найбільш цінні для студентів, а де вони зупиняються.

AdminPanel є повнофункціональною панеллю управління, організованою у вигляді вкладок. Вкладка статистики відображає зведені показники платформи: кількість користувачів, курсів, записів, загальний дохід та відсоток завершення. Підтримується фільтрація статистики за часовим діапазоном з попередньо заданими пресетами (тиждень, місяць, рік) та довільним вибором дат. Графіки динаміки реєстрацій та доходу відображаються з адаптивною гранулярністю: по днях для короткого діапазону та по місяцях для довгого. Вкладка користувачів надає пошук з фільтрацією за роллю, можливість зміни ролі, блокування та розблокування з обов'язковим зазначенням причини. Вкладка курсів дозволяє переглядати деталі будь-якого курсу, змінювати статус публікації та видаляти. Вкладка

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 58   |

викладачів відображає рейтинг за доходом з повною статистикою по кожному. Вкладка журналу аудиту відображає хронологічний список усіх адміністративних дій.

Весь інтерфейс реалізований із кастомними CSS-стилями без використання UI-фреймворків. Цей підхід забезпечує максимальний контроль над зовнішнім виглядом і виключає залежність від специфіки сторони бібліотек. CSS-змінні визначені для всіх ключових кольорів, розмірів тіней та радіусів. Система кольорів базується на нейтральних відтінках; основні кольорові змінні --accent визначені як #0a0a0a (світла тема) та #f0f0f0 (темна тема). Колір #4f46e5 використовується локально в окремих елементах інтерфейсу.

Темна та світла теми реалізовані через дві групи CSS-змінних, що перемикаються через додавання класу dark до кореневого елемента<html>. ThemeContext зберігає поточну тему в localStorage і синхронізує її між вкладками через подію storage. Жовті акцентні іконки збережені в обох темах як стабільний елемент брендингу.

Адаптивна верстка реалізована через CSS Grid та Flexbox із медіа-запитами для трьох розмірів: мобільного (до 768px), планшетного (до 1024px) та десктопного (більше 1024px). На мобільних пристроях навчальний план каталогу та фільтри прибираються з основного контенту та відображаються у виїзних панелях (drawer), що відкриваються через кнопки. На мобільній версії сторінки курсу відео-плеєр розміщується над навчальним планом, а не поруч.

Компонент Toast відображає тимчасові сповіщення у правому нижньому куті екрана з автоматичним зникненням через 3 секунди. Підтримуються типи: успіх, помилка та загальне інформаційне повідомлення. Компонент Skeleton із CSS-анімацією shimmer відображає заглушки під час завантаження даних. ProfilePage дозволяє редагувати ім'я та аватар, а також змінювати пароль через окрему форму з підтвердженням поточного пароля.

Проект повністю контейнеризовано за допомогою Docker для забезпечення відтворюваного середовища на всіх етапах розробки та

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 59   |

розгортання. Окремі Dockerfile реалізовані для серверної та клієнтської частин із застосуванням багатоетапної збірки (multi-stage build). Для клієнтської частини перший етап виконує збірку статичних файлів через Node.js[21], другий - копіює результат в образ nginx для ефективного роздавання статички. Такий підхід зменшує розмір фінального образу та розділяє середовища збірки та виконання.

Docker Compose оркеструє три сервіси: postgres з PostgreSQL 16, backend з NestJS API та frontend з React-додатком за nginx. Між сервісами налаштовані залежності: backend запускається лише після готовності postgres. Мережа Docker забезпечує ізоляцію між контейнерами та надає доступ лише до задекларованих портів. Конфігурація зовнішніх сервісів (WayForPay, Nodemailer, JWT-секрети, URL фронтенду та бекенду) передається через змінні середовища з файлу .env, що забезпечує безпеку секретних даних і простоту налаштування для різних середовищ (розробка, staging, продакшн).

Система розгортається на платформі Render з автоматичним CI/CD на основі GitHub-репозиторію. При кожному push до основної гілки автоматично запускається збірка та розгортання нової версії. PostgreSQL надається як керована база даних Render, що забезпечує автоматичне резервне копіювання та моніторинг.

## Висновки до другого розділу

У другому розділі виконано комплексне проєктування та реалізацію вебплатформи онлайн-навчання - повнофункціонального full-stack рішення на основі React з TypeScript та NestJS з PostgreSQL. Представлена робота охоплює всі рівні системи: від формалізованих функціональних вимог до конкретних рішень із реалізації.

На основі детального аналізу функціональних вимог визначено п'ять ролей користувачів - студент, викладач, модератор, адміністратор і супер-адміністратор - та повний перелік варіантів використання системи. Варіанти використання охоплюють: реєстрацію та автентифікацію через email/пароль і

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 60   |

Google OAuth, управління навчальним контентом з ієрархічною структурою "модуль → урок", відстеження прогресу на рівні окремих уроків з урахуванням переглянутих секунд відео, платіжну взаємодію через WayForPay, систему підписок, промокодів, Q&A, відгуків, вішлісту, реферальних програм, виплат викладачам та аудиту адміністративних дій.

Розроблена об'єктно-орієнтована модель охоплює 17 основних сутностей: User, Course, CourseModule, Lesson, Enrollment, Progress, Payment, Certificate, Subscription, PromoCode, Review, QaQuestion, QaAnswer, Notification, AuditLog, Referral, PayoutRequest. Між сутностями встановлено розгалужену мережу зв'язків, що коректно відображає реальну предметну область. Схема бази даних еволюційно формується через 15 TypeORM-міграцій, що забезпечує контрольоване оновлення.

Архітектура взаємодії клієнта з сервером базується на REST API з автоматично згенерованою Swagger-документацією, декларативній валідації вхідних даних через class-validator та JWT-автентифікації з access/refresh-механізмом. Типізована система моделей даних через TypeScript усуває цілий клас помилок ще на етапі компіляції. Клієнтська частина використовує React Context API для глобального управління станом та кастомні хуки для інкапсуляції логіки взаємодії з API.

Реалізовані алгоритми охоплюють усі ключові процеси платформи. JWT-автентифікація забезпечує безпечне управління сесіями з підтримкою Google OAuth та скидання пароля через email. Послідовне відстеження прогресу уроків через upsert-операції з принципом максимуму гарантує коректність даних при паралельних запитах. Двоетапна обробка платежів WayForPay з верифікацією HMAC-MD5 підпису та автоматичним наданням доступу забезпечує надійність фінансових операцій. Формування PDF-сертифікатів через pdftkit після підтвердження 100-відсоткового прогресу і зберігання у форматі Base64 в базі даних забезпечує незалежність від зовнішніх сховищ. Глобальний аудит адміністративних дій через NestJS-

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 61   |

інтерсептор і захист від перевантаження через ThrottlerGuard забезпечують безпеку та надійність платформи.

Реалізація функціоналу базується на перевірених сучасних підходах: NestJS-модулі для ізоляції предметних областей, TypeORM для типобезпечної роботи з PostgreSQL, WayForPay для обробки платежів у гривнях, повна контейнеризація через Docker Compose для відтворюваного розгортання. Кастомна CSS-система з підтримкою темної теми та адаптивна верстка для мобільних, планшетних та десктопних пристроїв забезпечують якісний користувацький досвід на всіх типах пристроїв.

Таким чином, у результаті виконання другого розділу розроблено та реалізовано цілісну вебплатформу онлайн-навчання, що об'єднує навчальний процес, системи оцінювання, фінансові механізми та адміністративний контроль в єдиному інтегрованому середовищі. Отримані результати є повністю функціональним програмним продуктом, готовим до розгортання та дослідного використання.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 62   |

## РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ОНЛАЙН-НАВЧАННЯ

### 3.1 Порядок встановлення та налаштування параметрів платформи

Для забезпечення публічного доступу до платформи та перевірки її роботи у реальних умовах було проведено розгортання у хмарі. Production-середовище використовує безкоштовні тарифи кількох незалежних провайдерів та не вимагає окремого серверного обладнання, що робить його придатним для демонстраційного проекту з мінімальними витратами на інфраструктуру.

Платформа розділена на три шари, кожен з яких розгортається на спеціалізованому хмарному сервісі. Розподіл відображає типову архітектуру сучасних веб-застосунків: статичний фронтенд на CDN, бекенд у контейнері та керована база даних. Кожен із провайдерів відповідає за свою задачу й може масштабуватися чи замінюватися окремо без впливу на інші шари.

Клієнтську частину побудовано на базі React з TypeScript і розгорнуто на Render - платформі-як-сервіс (PaaS) з автоматичним деплоєм з GitHub, що підтримує одночасний хостинг як статичного фронтенду, так і серверних Node.js-застосунків. Render автоматично виявляє тип проєкту, виконує збірку та надає публічний HTTPS-домен після першого push у репозиторій.

Серверну частину реалізовано на NestJS та розгорнуто на Render - платформі-як-сервіс, що нативно підтримує Node.js-застосунки. Render самостійно виявляє package.json у репозиторії й виконує стандартну послідовність команд `npm install`, `npm run build` та запуск отриманого артефакту. Цей підхід усуває потребу в ручному написанні Dockerfile і дозволяє публікувати оновлення одним push-ом в Git.

Базу даних PostgreSQL розгорнуто як окремий сервіс безпосередньо на Render. Платформа надає керований PostgreSQL-інстанс з автоматичним резервним копіюванням і захищеним підключенням через внутрішню мережу сервісів Render. TypeORM і pg-провайдер підключаються до бази через

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 63   |

DATABASE\_URL без жодних змін у коді відносно локального середовища розробки.

Архітектура production-розгортання платформи

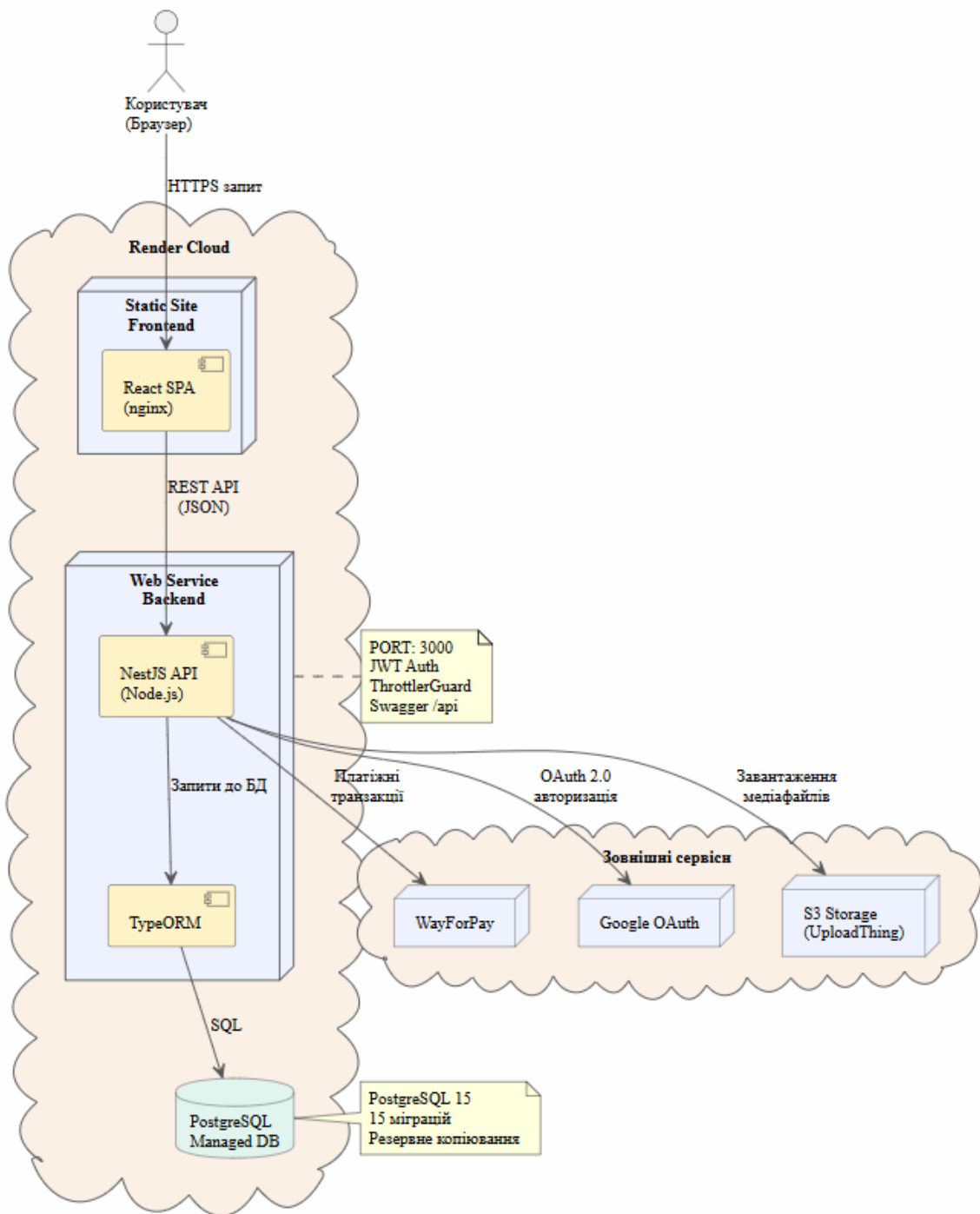


Рис. 3.1 – Архітектура production-розгортання платформи

Маршрут типового запиту до платформи проходить наступним чином. Браузер користувача звертається до публічного домену Render, на якому розгорнуто статичні файли React-застосунку. Клієнтський код виконує REST-

запит до серверного NestJS-додатку, розгорнутого як окремий Web Service на Render. Завдяки тому, що фронтенд і бекенд розгорнуто в межах одного середовища Render, CORS-конфігурація зводиться до зазначення FRONTEND\_URL у змінних оточення. NestJS-застосунок виконує перевірку JWT-токена, звертається до PostgreSQL через внутрішню мережу і повертає результат клієнту.

Налаштування платформи у production-середовищі повністю керується через змінні оточення, що відповідає підходу Twelve-Factor App[20]. Конфігураційні файли .env не використовуються у проді - замість них NestJS автоматично читає environment variables. Перелік змінних оточення наведено в таблиці 3.1.

Таблиця 3.1

Змінні оточення production-середовища

| Змінна оточення                           | Призначення                                                      |
|-------------------------------------------|------------------------------------------------------------------|
| DATABASE_URL                              | Рядок підключення до PostgreSQL-сервісу Render з параметрами SSL |
| JWT_SECRET                                | 64-символьний секретний ключ для підпису JWT-токенів             |
| JWT_EXPIRES_IN                            | Час життя access-токена (за замовчуванням 15 хвилин)             |
| REFRESH_TOKEN_EXPIRES_IN                  | Час життя refresh-токена (за замовчуванням 7 днів)               |
| GOOGLE_CLIENT_ID,<br>GOOGLE_CLIENT_SECRET | Ключі OAuth-додатку Google для авторизації через Google          |
| WAYFORPAY_MERCHANT_ACCOUNT                | Ідентифікатор мерчанта WayForPay для обробки платежів            |
| WAYFORPAY_SECRET_KEY                      | Секретний ключ для підпису HMAC-MD5 платіжних запитів            |

|              |                                                                                       |
|--------------|---------------------------------------------------------------------------------------|
| FRONTEND_URL | Дозволений CORS-дозвіл - домен фронтенду на Render (graduation-frontend.onrender.com) |
| NODE_ENV     | Режим роботи застосунку, встановлюється у значення production                         |

Процес першого розгортання складається з трьох кроків, які виконуються в межах єдиної платформи Render. Спочатку у дашборді Render створюється PostgreSQL-сервіс, після чого платформа автоматично генерує рядок підключення DATABASE\_URL. Далі створюється Web Service для бекенду з прив'язкою до GitHub-репозиторію та набором змінних оточення. Після першого запуску TypeORM автоматично застосовує міграції та заповнює базу даних початковими даними через seed-скрипт.

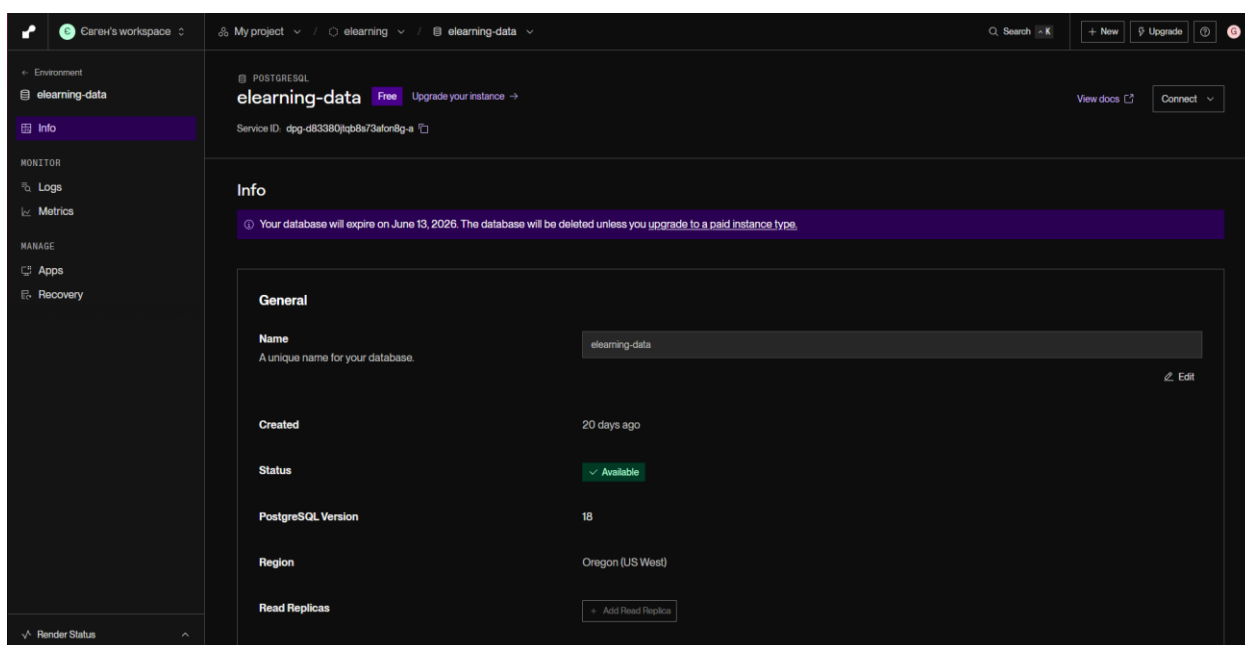


Рис. 3.2 – Дашборд Render з PostgreSQL-сервісом та підключеними Web Services

Далі на Render створюється Web Service з прив'язкою до GitHub-репозиторію, кореневим каталогом серверної частини і набором змінних оточення з таблиці 3.1. Render автоматично виявляє Node.js-проект і запускає збірку, після чого генерує публічний домен виду elearning-backend-hhfg.onrender.com, через який бекенд стає доступним для зовнішніх запитів.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 66   |

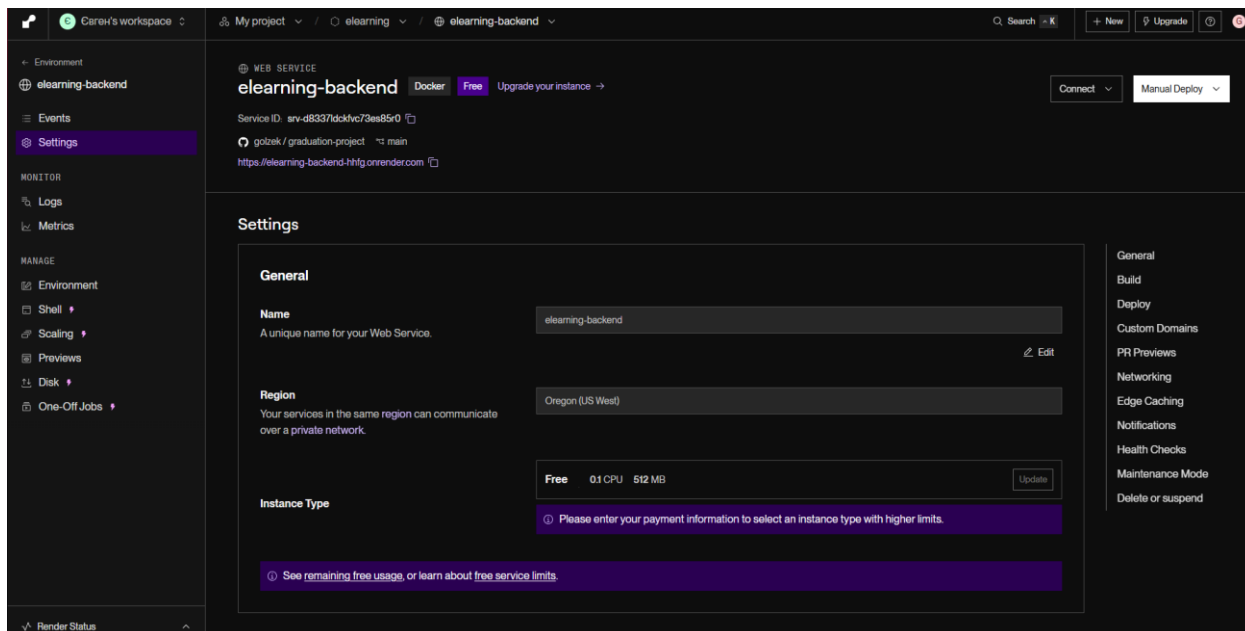


Рис. 3.3 – Web Service бекенду на Render зі статусом Live

Третім кроком на Render створюється Static Site для фронтенду: прив'язується той же GitHub-репозиторій, вказується кореневий каталог клієнтської частини. Render автоматично визначає React як використовуваний фреймворк, виконує `npm run build` і публікує вміст директорії `build`. Змінна оточення `REACT_APP_API_URL` вказує на домен бекенд-сервісу.

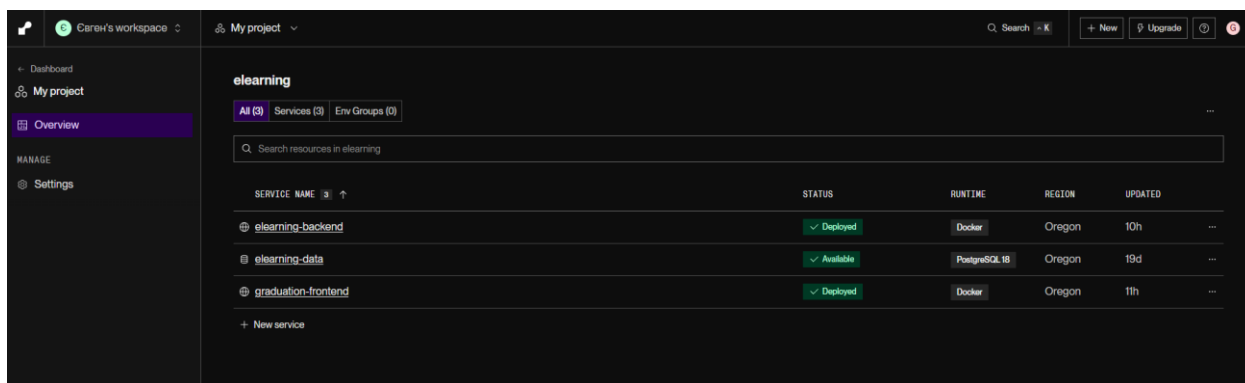


Рис. 3.4 – Дашборд Render зі статусом усіх сервісів платформи

Підтвердженням успішного розгортання слугують знімки робочого застосунку у production-середовищі. Жива головна сторінка платформи показана на рисунку 3.5, а адміністративна панель з даними з production-бази - на рисунку 3.6.

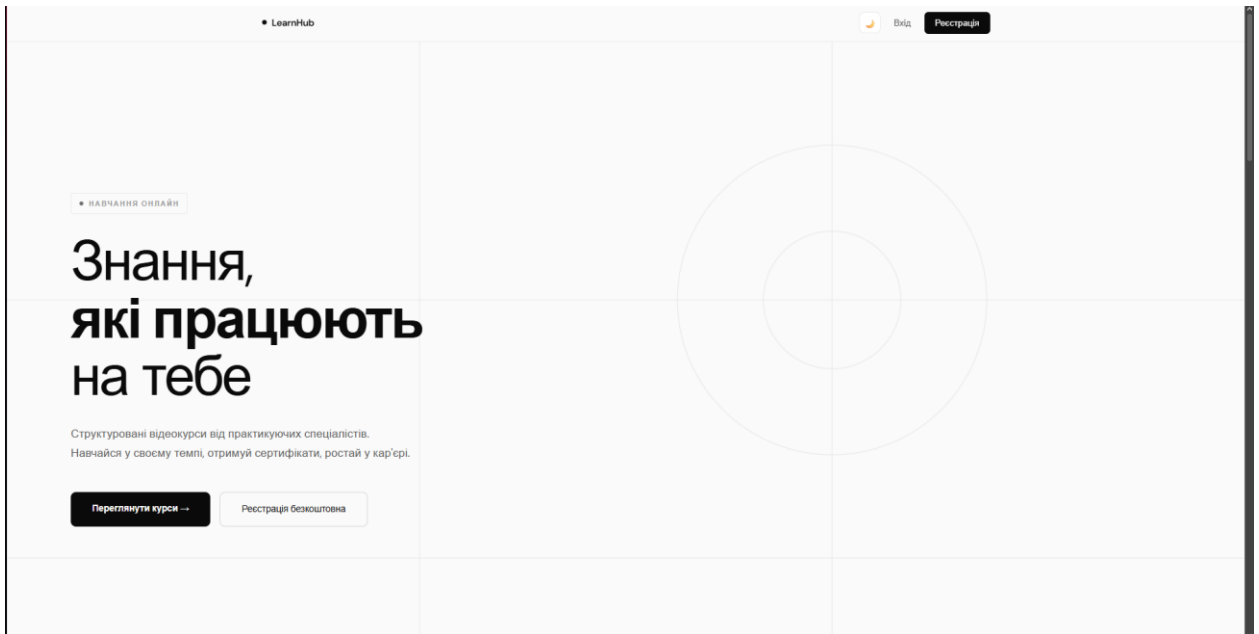


Рис. 3.5 – Головна сторінка платформи

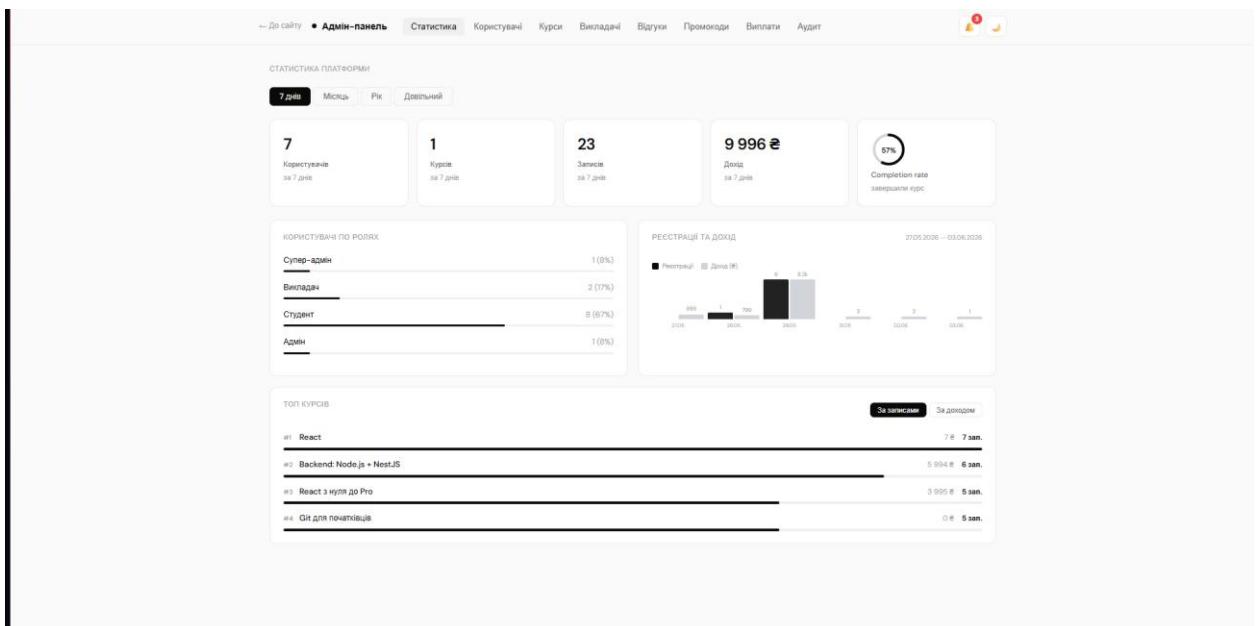


Рис. 3.6 – Адміністративна панель платформи у production-середовищі

Подальші оновлення коду розповсюджуються автоматично завдяки інтеграції GitHub-репозиторію з платформою Render: при кожному push у гілку main система автоматично ініціює збірку та розгортання нової версії сервісу без ручного втручання. Завдяки такому підходу платформа залишається доступною за публічним посиланням цілодобово, а відсутність власного серверного обладнання дозволяє суттєво знизити операційні витрати на підтримку інфраструктури.

### 3.2 Структура інтерфейсу вебплатформи онлайн-навчання

Інтерфейс платформи побудований як односторінковий React-застосунок (SPA) з єдиним layout-каркасом - навігаційною панеллю угорі, опціональним сайдбаром та основним змістом сторінки у центрі. Навігаційна панель містить логотип з посиланням на головну, навігаційні посилання (Головна, Курси, Про нас), перемикач теми (світла або темна), а для авторизованих користувачів - дзвоник сповіщень, ім'я користувача з посиланням на профіль, посилання на список бажаних та кнопку виходу. Сторінки входу і реєстрації відображаються поза цим каркасом, на чистому тлі, для максимальної концентрації уваги.

Перемикання теми виконується миттєво без перезавантаження сторінки. Тема зберігається у localStorage і застосовується через CSS-змінні на кореневому елементі. За замовчуванням використовується світла тема; вибір зберігається у localStorage і відновлюється при наступних візитах.

Дзвоник сповіщень у навігаційній панелі показує бейдж з кількістю непрочитаних повідомлень і за кліком відкриває випадаючий список з останніми записами - запис на курс, отримання сертифіката, зміна статусу курсу. Кожне сповіщення містить тип, короткий опис і час події. Нижче розглянуто основні екрани платформи у порядку звичайного користування авторизованим відвідувачем.

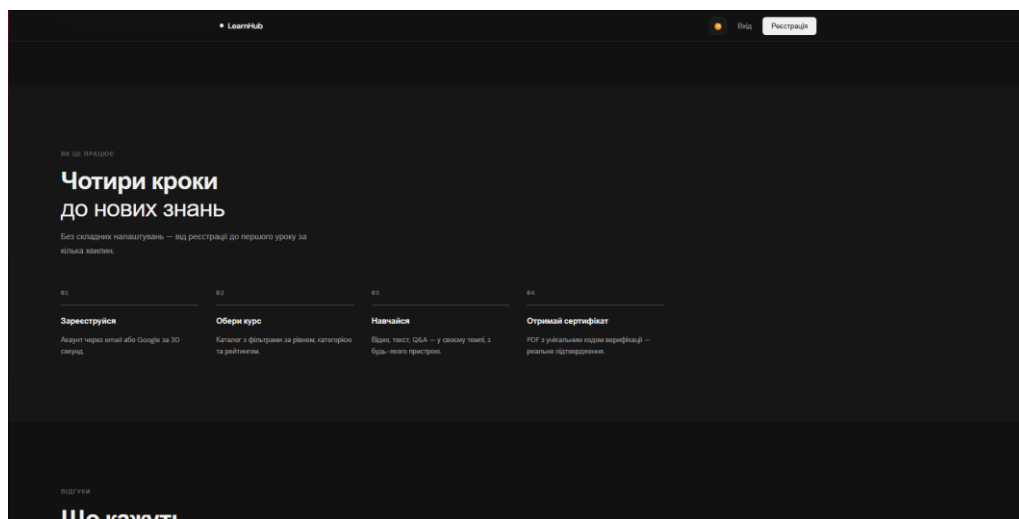


Рис. 3.7 – Головна сторінка платформи

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 69   |

Каталог курсів є центральною точкою навчального контенту (рис. 3.8). На сторінці відображено сітку карток із курсами, кожна картка містить мініатюру або автоматично згенерований градієнтний фон, назву, рейтинг у вигляді зірок, ім'я автора, рівень складності та ціну з позначкою FREE для безкоштовних курсів. Для записаних студентів відображається прогрес-бар із відсотком завершення. Угорі сторінки розташовано пошук за назвою та набори фільтрів - за категорією, рівнем складності, ціновим діапазоном та мінімальним рейтингом. Стан фільтрів синхронізується з URL-параметрами, що дозволяє ділитися посиланнями на відфільтровані результати.

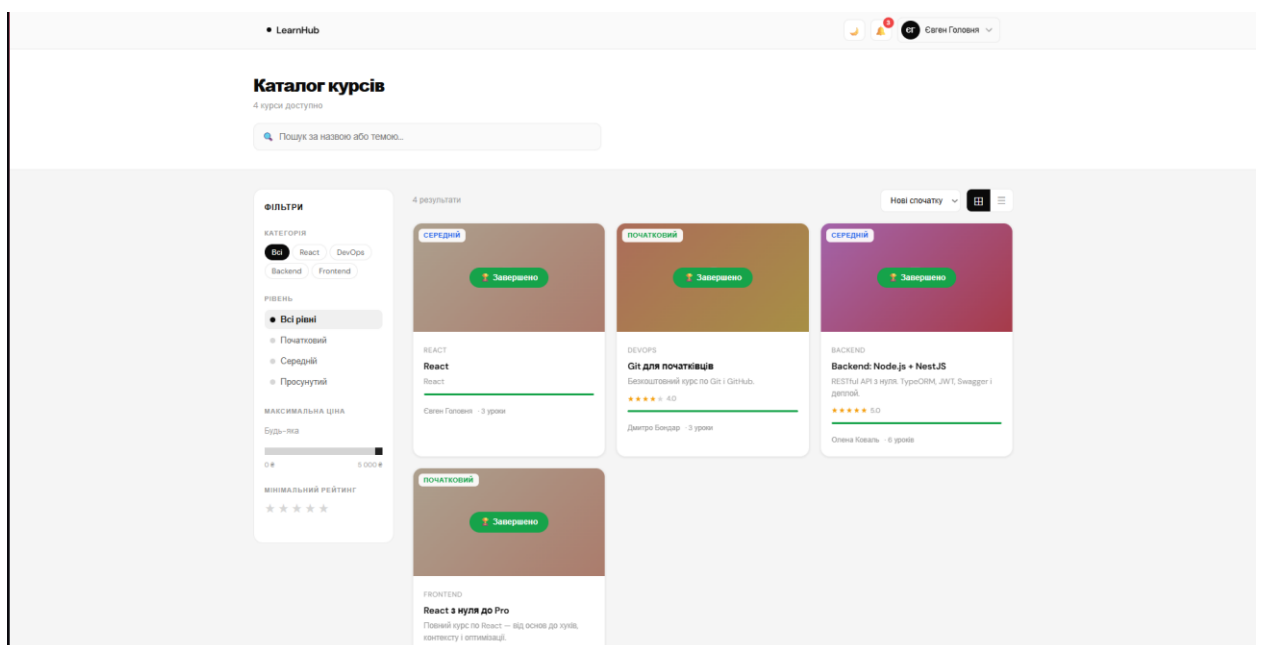


Рис. 3.8 – Каталог курсів з фільтрами та режимом сітки

Сторінка входу реалізована у вигляді мінімалістичної форми, розташованої у правій частині екрана, яка містить поля для введення email-адреси та пароля, кнопку входу через Google для швидкої OAuth-автентифікації, а також посилання на сторінку реєстрації для нових користувачів. Для підвищення зручності використання передбачено інформативні стани помилок: окреме повідомлення для випадку введення невірних облікових даних та окреме – для заблокованих акаунтів, що дозволяє користувачу чітко розуміти причину відмови у доступі та вжити відповідних заходів. (рис. 3.9).

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 70   |

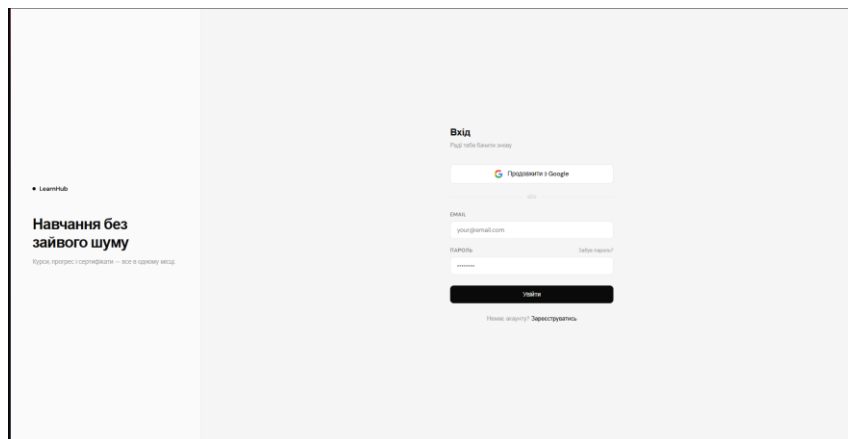


Рис. 3.9 – Сторінка входу з підтримкою Google OAuth

Сторінка реєстрації побудована аналогічно і містить поля ім'я, email, пароль (рис. 3.10). Валідація відбувається на стороні клієнта: пароль має містити щонайменше шість символів, email - відповідати стандартному формату. Адмінську роль через звичайну реєстрацію отримати неможливо.

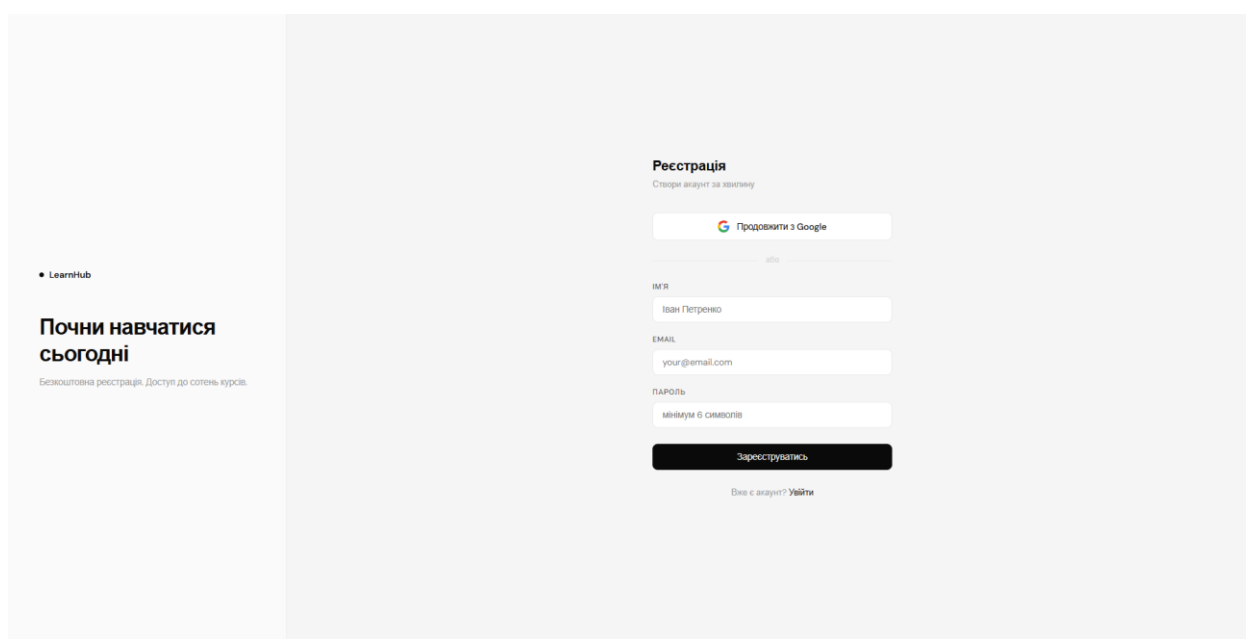


Рис. 3.10 – Сторінка реєстрації

Центральним елементом навчальної взаємодії є сторінка курсу. Сторінка має триколонкову структуру: хедер із загальною інформацією та карткою дії, ліва колонка з навчальним планом і права колонка з плеєром активного уроку (рис. 3.11). Картка дії є динамічною: для не записаних студентів відображається кнопка запису або компонент оплати, для записаних - прогрес-бар із відсотком завершення та кнопка продовження, при 100% завершенні – кнопка отримання сертифіката.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 71   |

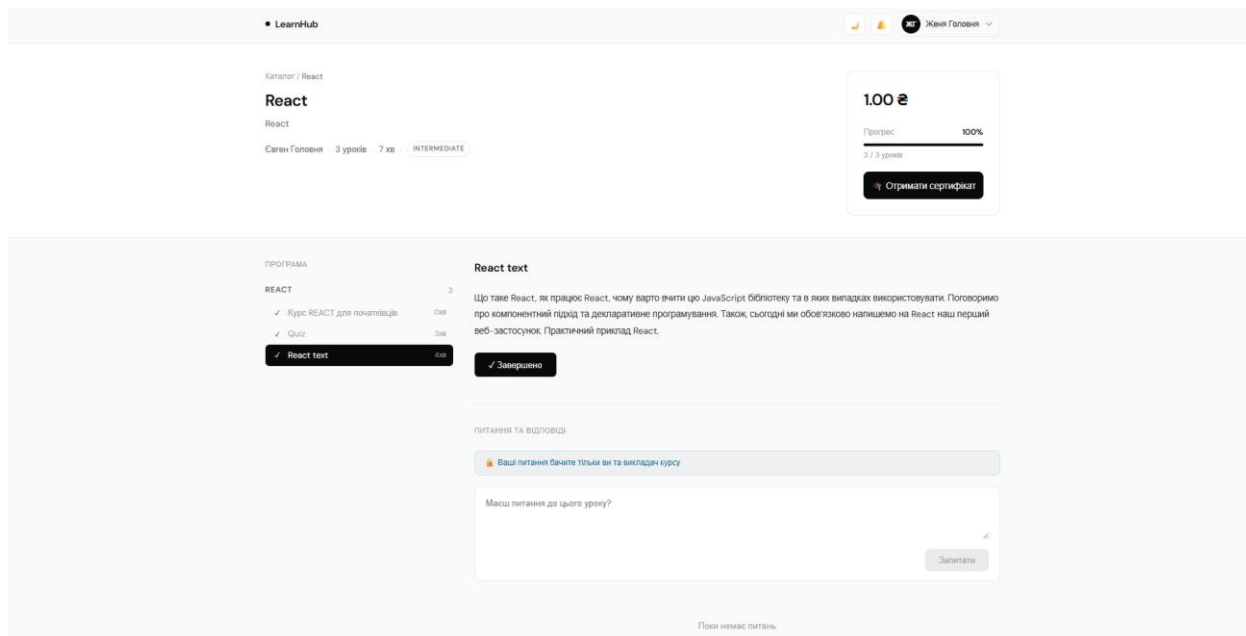


Рис. 3.11 – Сторінка курсу з навчальним планом та текстом

Ліва бічна панель відображає навчальний план у вигляді ієрархії модулів і уроків. Модулі можна розгортати та згорнути кліком по заголовку. Кожен урок відображає іконку типу (відео, текст або quiz), назву та тривалість. Заблоковані уроки відображаються з напівпрозорістю. Завершені уроки позначаються зеленою галочкою. При зміні активного уроку правий плеєр оновлюється відповідним контентом (рис. 3.12).

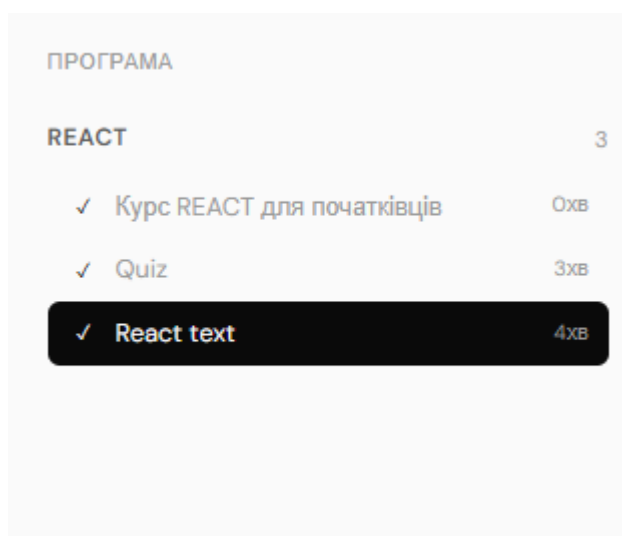


Рис. 3.12 – Навчальний план курсу з позначенням прогресу

Компонент LessonPlayer обробляє три типи контенту. Для відеоуроків використовується виключно YouTube: посилання зберігається у базі даних, а відео вбудовується через iframe з YouTube Embed API - такий підхід усуває

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 72   |

потребу у власному сховищі медіафайлів і забезпечує стабільну доставку відео. Завершення уроку підтверджується вручну через кнопку після перегляду. Для текстових уроків відображається HTML-контент, конвертований з Markdown (рис. 3.13). Для quiz-уроків компонент QuizPlayer відображає запитання з варіантами відповідей і перевіряє результат після відправки.

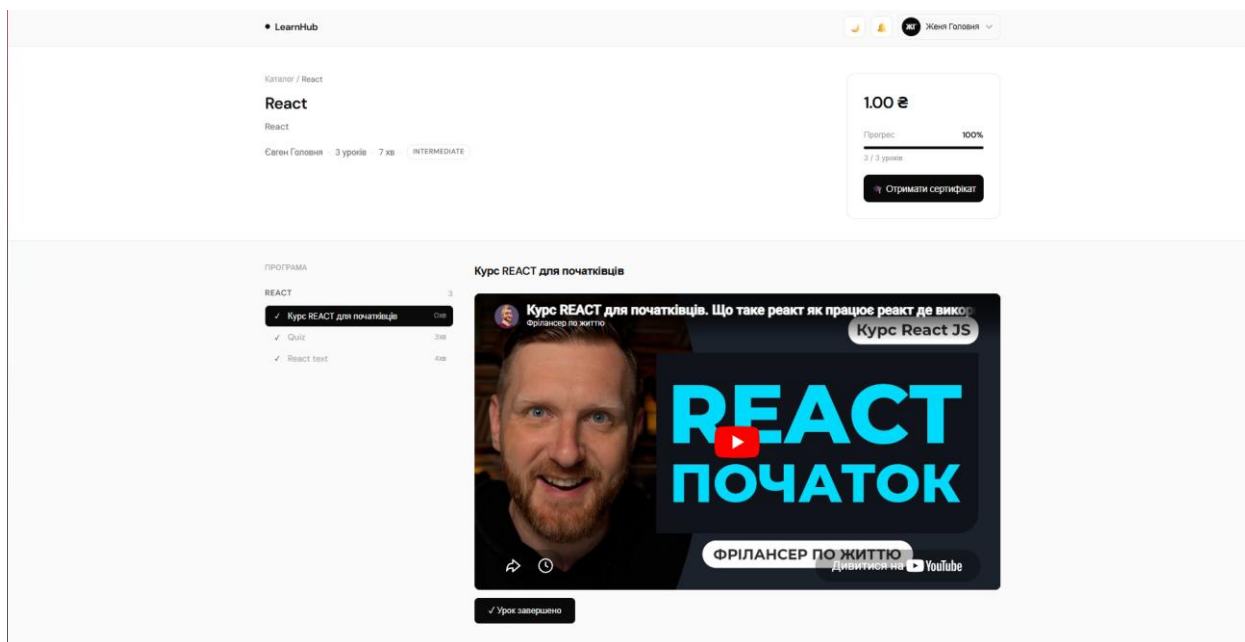


Рис. 3.13 – Плеєр відеоуроку зі статусом проходження

Після досягнення 100% прогресу система автоматично ініціює видачу сертифіката. Відображається модальне вікно з вітанням, кнопкою завантаження PDF та кодом верифікації, а також формою для залишення відгуку про курс (рис. 3.14).

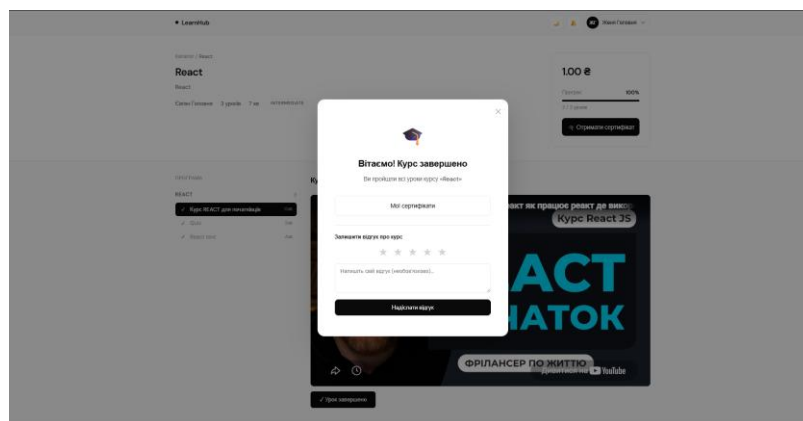


Рис. 3.14 – Модальне вікно отримання сертифіката після завершення курсу

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 73   |

Авторизований студент отримує доступ до особистого кабінету, де відображаються всі записані курси з поточним прогресом. Сторінка особистої аналітики показує загальний час перегляду, поточну серію активних днів, розподіл активності за годинами та днями тижня у вигляді теплової карти (рис. 3.15).

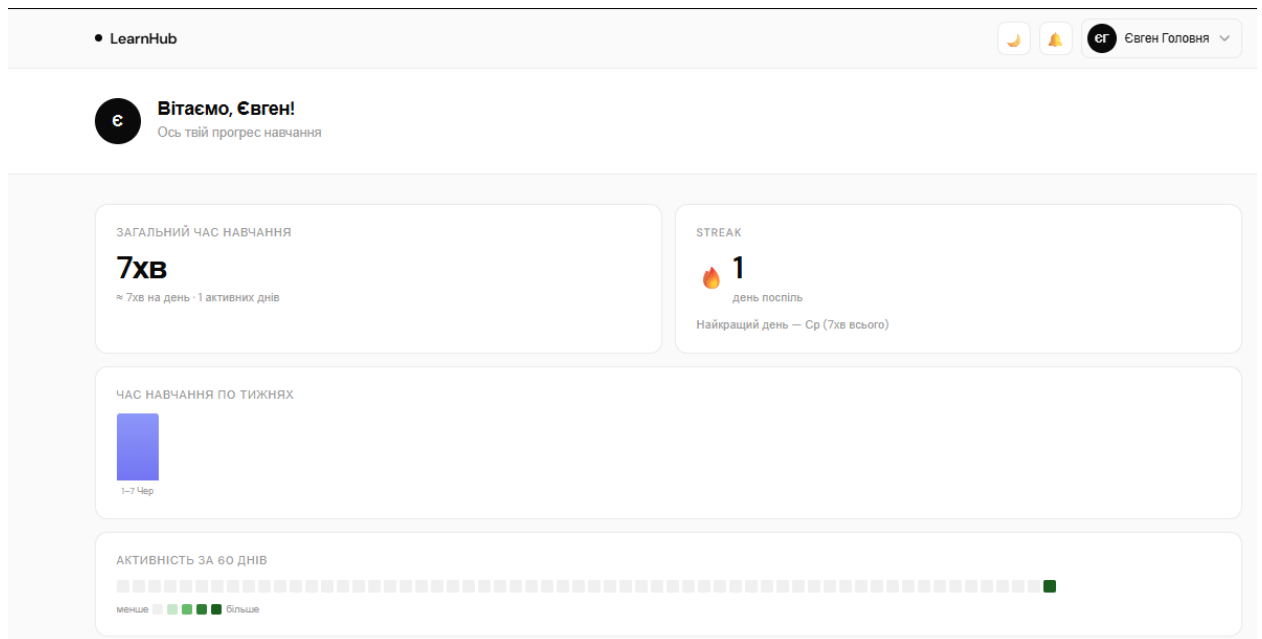


Рис. 3.15 – Особиста аналітика студента з тепловою картою активності

Сторінка вішліста зберігає курси, що зацікавили студента. Кожна картка у списку бажаних відображає назву, автора, ціну та кнопки «Записатися» та «Видалити зі списку». Стан вішліста синхронізується між сторінками через контекст і зберігається на сервері, тому не зникає після перезавантаження сторінки.

Сторінка профілю надає користувачу зручні інструменти для редагування особистих даних та керування акаунтом. (рис. 3.16). Передбачено можливість завантаження аватара у форматах JPEG, PNG або WebP, а також редагування імені, прізвища та короткої біографії. Окремо виділено секцію перегляду поточної підписки з кнопкою управління нею, а в нижній частині сторінки розміщено блок відповідності вимогам GDPR із можливістю повного видалення акаунту.



курсів зі статусами та швидкими діями - переходом до редагування та аналітики. Форма створення та редагування курсу дозволяє заповнити базову інформацію, завантажити обкладинку і управляти ієрархічною структурою модулів і уроків.

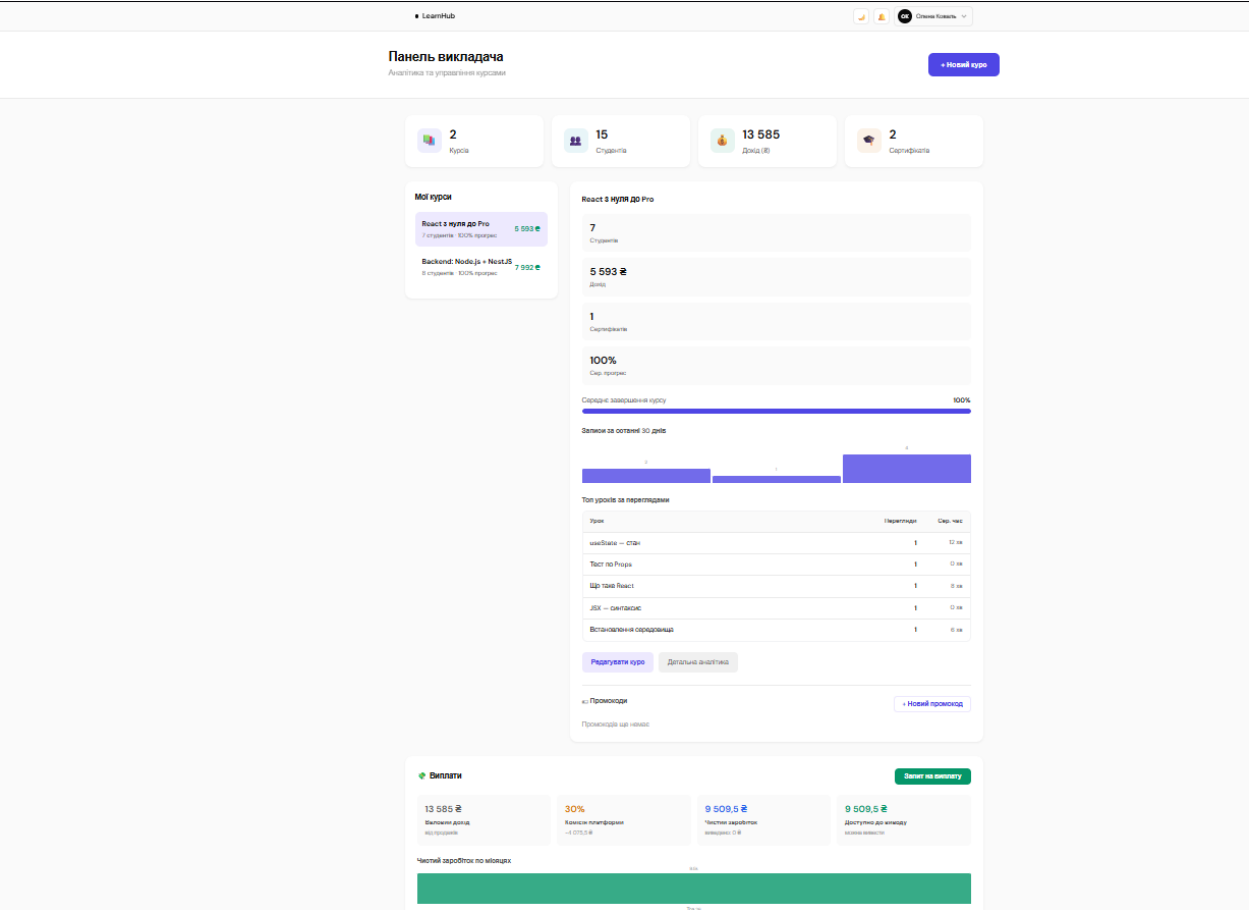


Рис. 3.18 – Дашборд викладача з загальною статистикою

Сторінка аналітики курсу надає викладачу комплексний огляд ключових показників: динаміка записів за останні 30 днів відображається у вигляді графіка, що дозволяє відстежувати тенденції зростання аудиторії, а середній відсоток прогресу студентів дає змогу оцінити загальний рівень залученості до навчання. Додатково представлено список найпопулярніших уроків та рейтинг відгуків, що допомагає викладачу визначати найефективніший контент і своєчасно реагувати на зворотний зв'язок від студентів. (рис. 3.19).

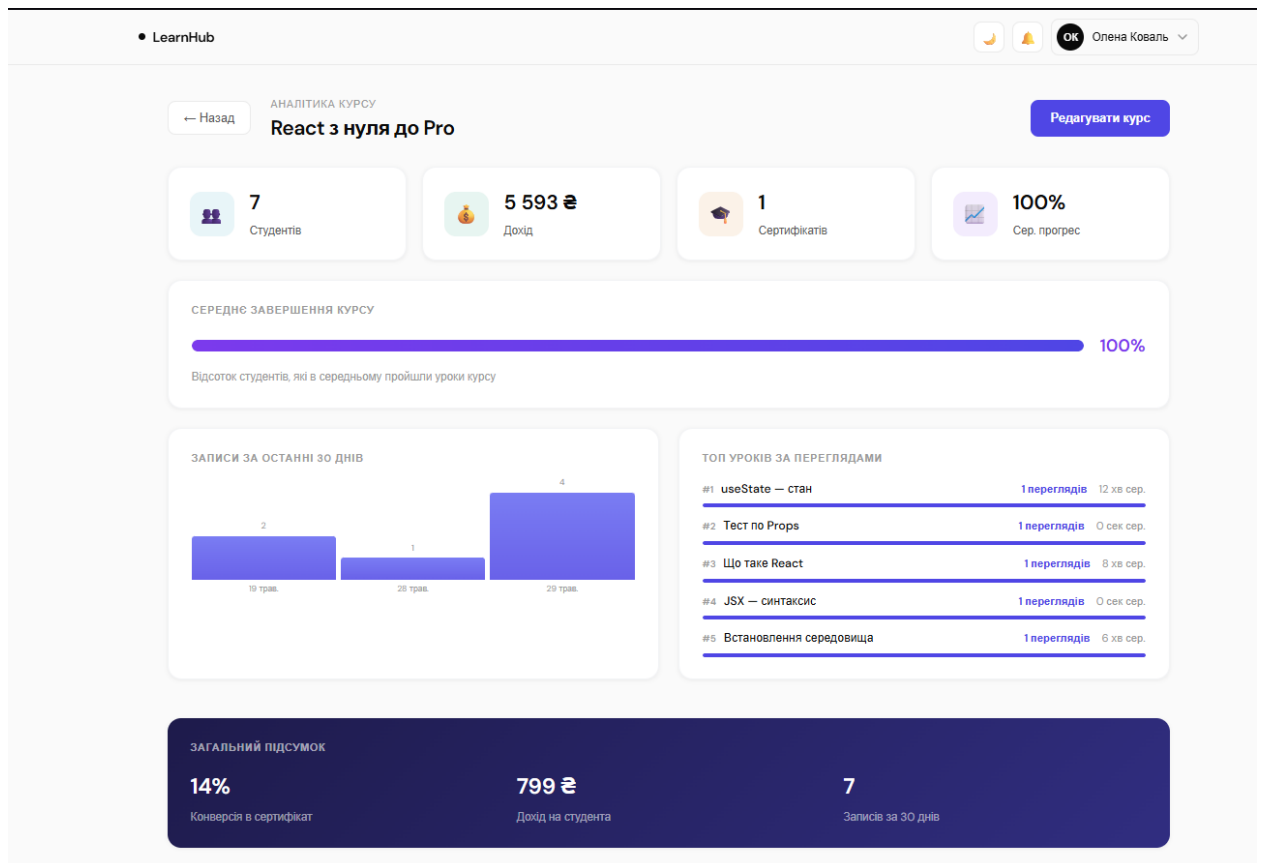


Рис. 3.19 – Аналітика курсу для викладача

Адміністративна панель доступна лише користувачам з роллю ADMIN або SUPER\_ADMIN і організована у вигляді вкладок (рис. 3.20). Вкладка статистики відображає ключові показники платформи з можливістю фільтрації за часовим діапазоном та графіками динаміки реєстрацій і доходу. Вкладка користувачів надає пошук за роллю, можливість зміни ролі, блокування та розблокування з обов'язковим зазначенням причини.

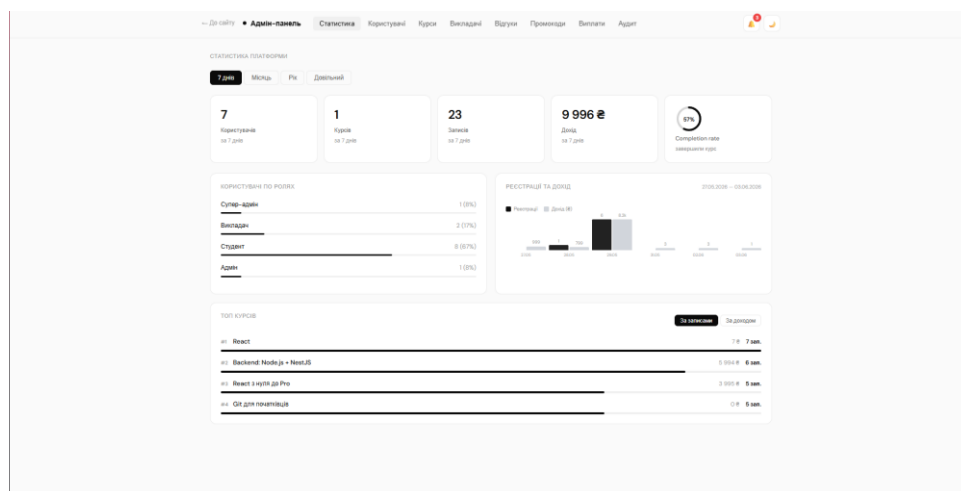


Рис. 3.20 – Адміністративна панель: загальна статистика платформи

Вкладка курсів дозволяє переглядати деталі будь-якого курсу та змінювати статус публікації - переводити курс зі стану DRAFT у PUBLISHED або ARCHIVED (рис. 3.21). Вкладка журналу аудиту відображає хронологічний список усіх адміністративних дій з HTTP-методом, маршрутом, ідентифікатором адміністратора та часом дії (рис. 3.22).

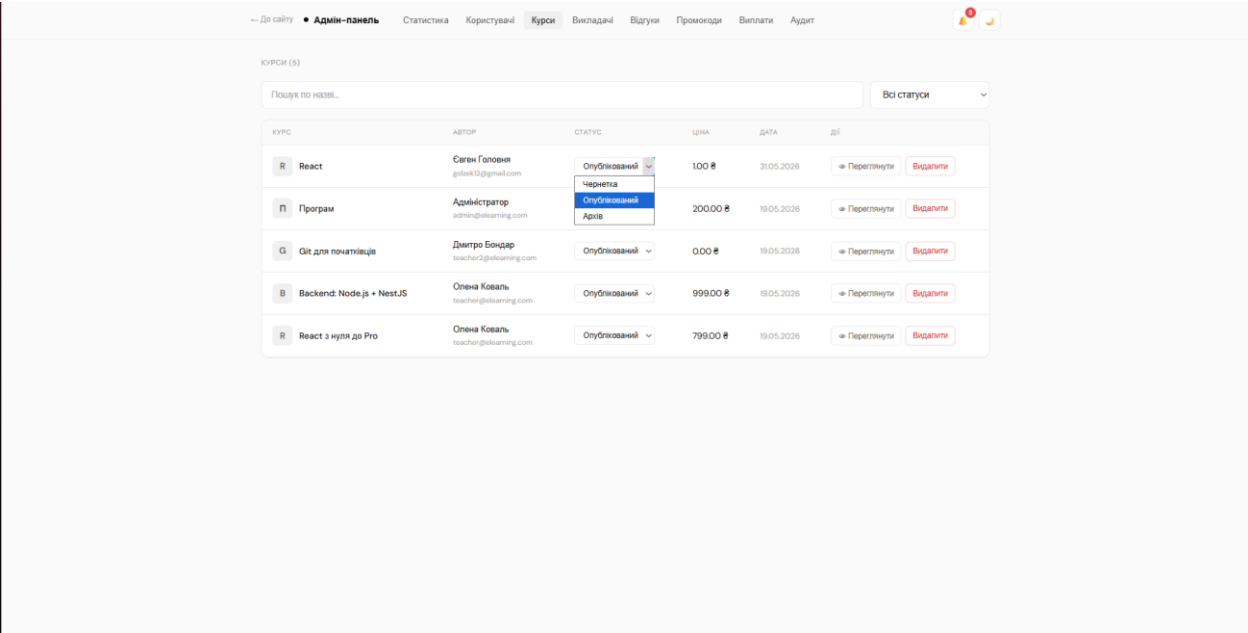


Рис. 3.21 – Адміністративна панель: модерація курсів

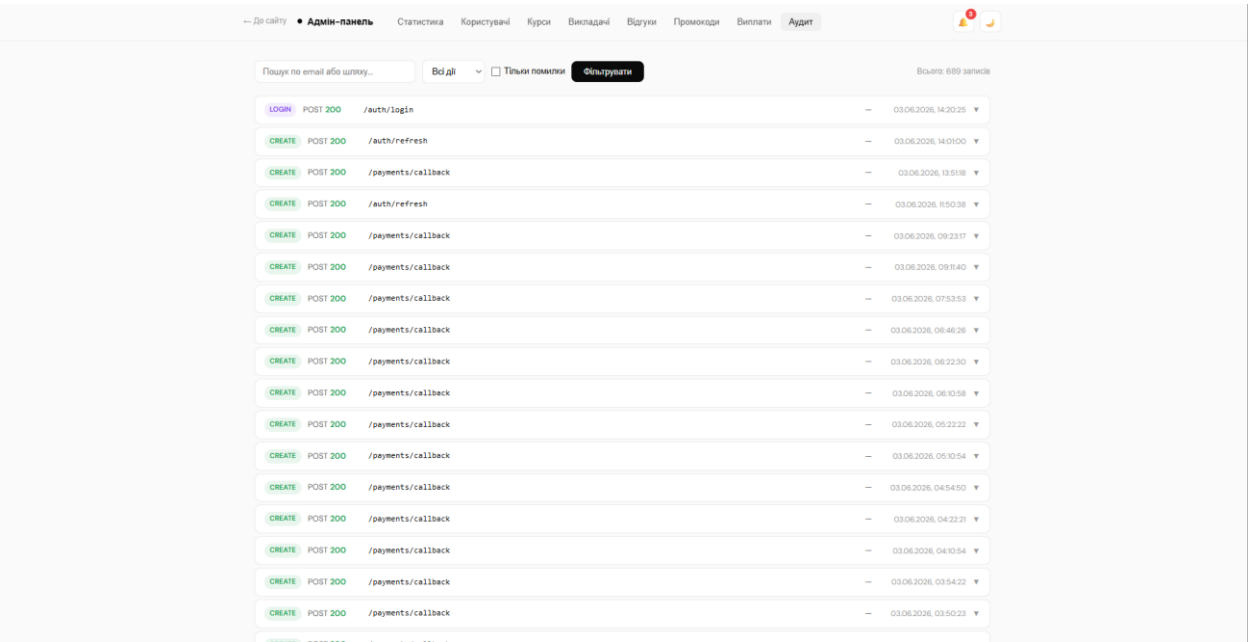


Рис. 3.22 – Адміністративна панель: журнал адміністративних дій

Описаний інтерфейс охоплює всі ключові сценарії використання платформи з боку трьох типів користувачів - студента, викладача і адміністратора - і дотримується принципів послідовності й мінімалізму: ті самі

візуальні шаблони повторюються на різних сторінках, що зменшує когнітивне навантаження і пришвидшує освоєння системи новим користувачем.

### 3.3 Тестування платформи

Тестування розробленої платформи проводилося безпосередньо на production-розгортанні з реальною базою даних і реальними мережевими затримками, що відображає умови експлуатації застосунку кінцевими користувачами. Свідомо було обрано підхід інтеграційного й end-to-end тестування замість виключно модульного: для діапазону задач, які виконує платформа, перевірка коректності окремих методів у відриві від їхнього контексту не дає реальної гарантії того, що система працює як єдине ціле.

Перевірка платформи розділена на чотири блоки. Перший блок перевіряє коректність роботи всіх функціональних модулів - автентифікації, каталогу курсів, навчального процесу та платіжної системи. Другий блок верифікує алгоритми оцінювання прогресу на крайових випадках. Третій блок оцінює продуктивність застосунку у production-середовищі за допомогою аудиту Google Lighthouse[15]. Четвертий блок підтверджує сумісність з основними браузерними рушіями шляхом виконання сценаріїв smoke-тестування у трьох незалежних engine.

Першим напрямком тестування стала перевірка коректності роботи функціональних сценаріїв платформи. Для кожного з ключових модулів було розроблено набір тестових сценаріїв, що перевіряють поведінку системи на коректних вхідних даних і на граничних випадках. Результати тестування функціональних сценаріїв наведено у таблиці 3.2.

Таблиця 3.2

#### Результати тестування функціональних сценаріїв

| Модуль         | Тестовий сценарій               | Очікуваний результат | Статус |
|----------------|---------------------------------|----------------------|--------|
| Автентифікація | Реєстрація через email і пароль | Акаунт створено      | +      |
| Автентифікація | Вхід через Google OAuth         | JWT-токен видано     | +      |

Продовження таблиці 3.2

|                   |                                          |                         |   |
|-------------------|------------------------------------------|-------------------------|---|
| Автентифікація    | Скидання пароля через email              | Пароль змінено          | + |
| Автентифікація    | Вхід із заблокованим акаунтом            | Помилка з причиною      | + |
| Каталог курсів    | Фільтрація за категорією, рівнем і ціною | Коректні результати     | + |
| Каталог курсів    | Синхронізація фільтрів з URL-параметрами | URL оновлено            | + |
| Навчальний процес | Запис на безкоштовний курс               | Доступ надано           | + |
| Навчальний процес | Автофіксація відео при перегляді 80%     | Урок завершено          | + |
| Навчальний процес | Видача сертифіката при 100% прогресу     | PDF створено            | + |
| Оплата            | Ініціація оплати через WayForPay         | Перенаправлення на шлюз | + |
| Оплата            | Верифікація підпису HMAC-MD5 у callback  | Доступ надано           | + |
| Q&A та відгуки    | Публікація питання та відповіді до уроку | Запис збережено         | + |

Усі дванадцять сценаріїв пройшли успішно. Особливої уваги заслуговує тестування платіжного модуля: верифікація підпису HMAC-MD5 коректно відхиляє підроблені callback-повідомлення і правильно нараховує доступ лише при статусі Approved від WayForPay. Алгоритм видачі сертифіката коректно перевіряє 100-відсоткове завершення і генерує PDF з підтримкою кирилиці через вбудовані TTF-шрифти.

Другим напрямком тестування стала верифікація алгоритму обрахунку прогресу проходження курсу, реалізованого у CoursesService. Для цього розроблено набір крайових тестових випадків, що охоплюють граничні ситуації відстеження прогресу.

Кожен тест виконує повний цикл: запис на курс, збереження прогресу з різними параметрами, отримання підсумкового відсотка. Результати наведено у таблиці 3.3.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 80   |

Таблиця 3.3

## Крайові випадки алгоритму відстеження прогресу

| Код   | Вхідні умови                                                    | Очікуваний відсоток | Фактичний відсоток | Статус |
|-------|-----------------------------------------------------------------|---------------------|--------------------|--------|
| PRG-1 | Жоден урок не переглянуто                                       | 0%                  | 0%                 | +      |
| PRG-2 | Переглянуто 1 з 2 уроків                                        | 50%                 | 50%                | +      |
| PRG-3 | Усі уроки завершено                                             | 100%                | 100%               | +      |
| PRG-4 | Відео переглянуто 79% - автофіксація не спрацювала              | 0%                  | 0%                 | +      |
| PRG-5 | Відео переглянуто 80% - автофіксація спрацювала                 | 100%                | 100%               | +      |
| PRG-6 | Спроба зменшити переглянуті секунди (нове значення < збережене) | Без змін            | Без змін           | +      |
| PRG-7 | Quiz: правильних відповідей < 50%                               | Незараховано        | Незараховано       | +      |
| PRG-8 | Quiz: правильних відповідей $\geq$ 50%                          | Зараховано          | Зараховано         | +      |

Усі вісім випадків пройшли з нульовою розбіжністю між очікуваним і фактичним результатом. Особливо важливим є PRG-6, який підтверджує коректність принципу максимуму при збереженні переглянутих секунд: повторний запит з меншим значенням не зменшує прогрес, що захищає від помилок відстеження на клієнті.

Третім напрямком тестування стала оцінка продуктивності клієнтської частини. Для виміру використано офіційний інструмент Google Lighthouse у режимі desktop, який обчислює час до першого змістовного відображення, час до інтерактивності, сумарне блокування основного потоку та низку інших ключових показників. Тестування проведено проти production-домену <https://graduation-frontend.onrender.com/courses> у двох режимах - Комп'ютер та Мобільний. Отримані оцінки показано на рисунку 3.23.

Звіт від 3 черв. 2026 р., 14:01:56

<https://graduation-frontend.onrender.com/courses>

Аналізувати



Мобільний



Комп'ютер



Дізнайтеся, як відбувається взаємодія в реальних користувачів

Немає даних



Виявляйте проблеми з продуктивністю

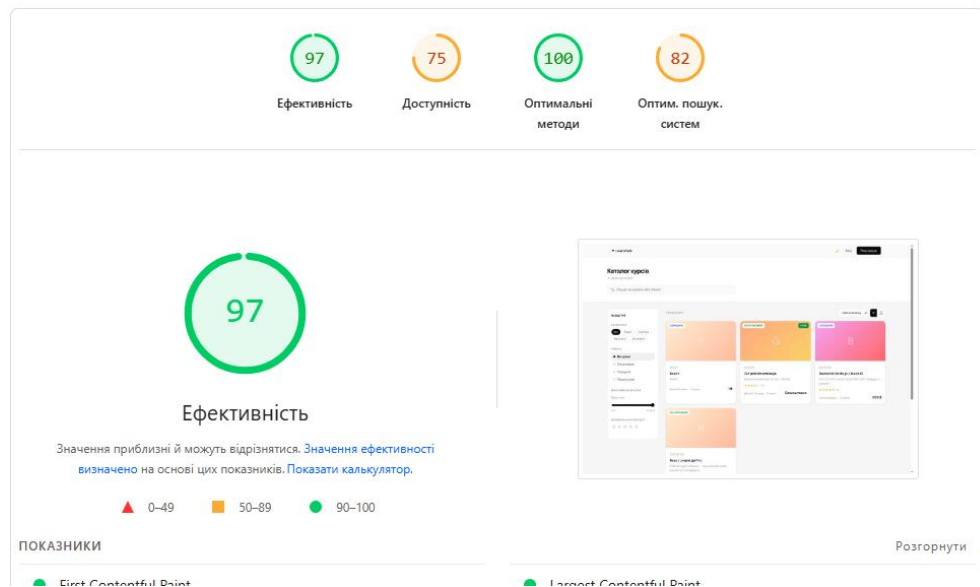


Рис. 3.23 – Підсумкові оцінки Lighthouse для production-розгортання

У режимі Комп'ютер платформа отримала 97/100 за Ефективністю, 75/100 за Доступністю, 100/100 за Оптимальними методами та 82/100 за SEO. У режимі Мобільний показники склали 93/100, 87/100, 100/100 та 82/100 відповідно. Оцінка Ефективності 97 для десктопу та 93 для мобільних пристроїв свідчить про відповідність вимогам до швидкості завантаження. Категорія Оптимальні методи з максимальним балом підтверджує дотримання сучасних стандартів веб-розробки - відсутність застарілих API, коректне налаштування HTTPS та відповідність вимогам безпеки браузера.

Швидкість відповіді серверного API виміряна з production-середовища. Час відповіді типових REST-запитів - отримання каталогу курсів, автентифікація, збереження прогресу - коливається в межах 500–2000 мс. Підвищена затримка пов'язана з особливістю безкоштовного тарифу Render: при відсутності трафіку понад 15 хвилин сервіс автоматично призупиняється і

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 82   |

перший запит після «сну» займає до 30 секунд на cold start. Для production-демонстрації такий рівень є прийнятним, а для реального впровадження усувається переходом на платний тариф або використанням stop-запитів для підтримання активності.

Четвертим напрямком тестування стала перевірка сумісності з різними браузерними рушіями. Було розроблено smoke-тест, що послідовно проходить п'ять ключових сценаріїв - завантаження головної сторінки, відкриття каталогу курсів, рендеринг відеоплеєра, автентифікація та перехід до адмін-панелі. Результати кросбраузерного тестування наведено у таблиці 3.4.

Таблиця 3.4

Результати кросбраузерного тестування

| Рушій        | Браузер      | Головна | Каталог | Плеєр | Логін | Адмін |
|--------------|--------------|---------|---------|-------|-------|-------|
| Chromium 141 | Chrome, Edge | +       | +       | +     | +     | +     |
| WebKit 26.4  | Safari 17+   | +       | +       | +     | +     | +     |
| Gecko 131    | Firefox      | +       | +       | +     | +     | +     |

Усі п'ять сценаріїв успішно пройшли в усіх трьох браузерних рушіях, що покривають практично весь сучасний ринок десктопних і мобільних браузерів. Відеоплеєр на базі нативного HTML5 та вбудованого YouTube iframe працює однаково на WebGL-стеці Chrome/Edge, на власній реалізації WebKit для Safari і на Gecko-рушії Firefox. Завдяки тому, що в кодовій базі не використовуються пропрієтарні API окремих браузерів, додаткова робота з адаптацією виявилася непотрібною.

Підсумовуючи результати чотирьох блоків тестування: функціональні сценарії коректно відпрацьовують за всіма ключовими бізнес-процесами, алгоритми відстеження прогресу працюють згідно з документованою логікою на всіх крайових випадках, продуктивність відповідає максимальному рівню за метриками Google Lighthouse, а сумісність охоплює усі сучасні браузерні рушії. Це підтверджує готовність платформи до використання у навчальному процесі.

## Висновки до третього розділу

В даному розділі було наведено детальну схему розгортання вебплатформи онлайн-навчання. Кожен структурний елемент системи описаний, а також продемонстровано його взаємодію з іншими частинами платформи. Архітектура production-середовища на базі платформи Render, що об'єднує хостинг фронтенду, бекенду та бази даних в єдиній екосистемі, забезпечує цілодобову доступність без витрат на серверне обладнання.

Далі було надано опис інтерфейсу розробленого програмного продукту. Були розглянуті всі функціональні можливості додатку залежно від ролі користувача: студент отримує доступ до каталогу курсів, навчального процесу, сертифікатів та особистої аналітики; викладач - до повного циклу управління курсами та аналітики успішності студентів; адміністратор - до панелі управління користувачами, модерації курсів та журналу аудиту.

Окрім того, було проведено тестування реалізованої платформи у production-середовищі. Перевірено коректність роботи всіх функціональних модулів, верифіковано алгоритм відстеження прогресу на восьми крайових випадках, виміряно продуктивність клієнтської частини за метриками Google Lighthouse та підтверджено сумісність з трьома основними браузерними рушіями. Результати тестування підтверджують готовність платформи до використання у навчальному процесі.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 84   |

## ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було успішно розроблено повнофункціональну вебплатформу для онлайн-навчання, що поєднує навчальний контент, систему оцінювання знань та комерційну складову в єдиній цифровій екосистемі.

1. Проведений аналіз існуючих освітніх платформ дозволив визначити ключові недоліки наявних рішень та сформулювати вимоги до розроблюваної системи. Порівняльний огляд Moodle, Coursera та Udemу виявив специфічну орієнтацію кожної платформи на певний сегмент користувачів з характерними обмеженнями: застарілий інтерфейс і перевантажена навігація Moodle, обмежені можливості для викладачів у Coursera та відсутність стандартизації якості контенту на Udemу. Обґрунтовано обрано технологічний стек на базі React з TypeScript та NestJS, що забезпечує типобезпечність, модульність і масштабованість рішення. Вибір PostgreSQL як реляційної СУБД та TypeORM для взаємодії з даними обґрунтовано потребами у надійному та контрольованому управлінні схемою через міграції.

2. Здійснено комплексне проєктування та реалізацію платформи з детальною розробкою всіх архітектурних компонентів. Спроектована об'єктно-орієнтована модель охоплює 17 основних сутностей та визначає п'ять ролей користувачів - студент, викладач, модератор, адміністратор і супер-адміністратор. Схема бази даних еволюційно формується через 15 TypeORM-міграцій, що забезпечує контрольоване оновлення у виробничому середовищі. REST API з автоматично згенерованою Swagger-документацією, JWT-автентифікація з механізмом access/refresh токенів та підтримка Google OAuth забезпечують безпечну та зручну взаємодію клієнта з сервером.

3. Запропоновано комплекс алгоритмів, що забезпечують ключові бізнес-процеси платформи. Алгоритм автентифікації охоплює три сценарії - вхід через форму, Google OAuth та скидання пароля - кожен з власними механізмами захисту. Алгоритм управління навчальним процесом реалізує

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 85   |

відстеження прогресу на рівні окремих уроків через upsert-операції з принципом максимуму переглянутих секунд, що гарантує коректність даних навіть при паралельних запитах. Двоетапна обробка платежів через WayForPay з верифікацією HMAC-MD5 підпису забезпечує надійність фінансових транзакцій. Автоматичне формування PDF-сертифікатів через pdftkit після підтвердження стовідсоткового прогресу та зберігання у форматі Base64 в базі даних забезпечують незалежність від зовнішніх сховищ.

4. Підтверджено успішність технічної реалізації через комплексне тестування у production-середовищі. Усі 13 функціональних сценаріїв пройшли успішно; алгоритм відстеження прогресу верифіковано на 8 крайових випадках з нульовою розбіжністю між очікуваними та фактичними результатами. Аудит Google Lighthouse зафіксував показники продуктивності 97/100 для десктопу та 93/100 для мобільних пристроїв, категорія «Оптимальні методи» отримала максимальний бал 100/100. Кросбраузерне тестування підтвердило сумісність з рушіями Chromium, WebKit та Gecko на всіх п'яти перевірених сценаріях.

Розроблена платформа успішно вирішує ключові проблеми існуючих рішень через централізовану екосистему управління навчальним процесом, що об'єднує курси, тести, сертифікати та платежі. Адаптивний інтерфейс із підтримкою темної і світлої теми, побудований на кастомних CSS-стилях без залежності від сторонніх UI-фреймворків, забезпечує якісний користувацький досвід на мобільних, планшетних та десктопних пристроях. Повна контейнеризація через Docker Compose та автоматичний CI/CD на базі GitHub і платформи Render забезпечили цілодобову доступність системи без витрат на власне серверне обладнання.

Архітектурна гнучкість системи забезпечує широкі можливості для подальшого розширення функціональності: впровадження системи рекомендацій курсів на основі машинного навчання, розширення аналітичних інструментів для викладачів, додавання підтримки мобільних додатків на React Native та інтеграція з відеосервісами для проведення живих занять.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 86   |

Модульна структура серверної частини на NestJS і чітке розділення відповідальності в клієнтській частині спрощують процес підтримки та розвитку платформи.

Розроблене рішення демонструє високий рівень технічної реалізації, відповідає сучасним стандартам веб-розробки та створює надійну основу для подальшого комерційного впровадження з урахуванням потреб різних категорій користувачів.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 87   |

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React Documentation [Електронний ресурс] – Режим доступу: <https://react.dev/> (дата звернення: 05.06.2026).
2. NestJS Documentation [Електронний ресурс] – Режим доступу: <https://docs.nestjs.com/> (дата звернення: 05.06.2026).
3. TypeScript Handbook [Електронний ресурс] – Режим доступу: <https://www.typescriptlang.org/docs/> (дата звернення: 05.06.2026).
4. PostgreSQL 16 Documentation [Електронний ресурс] – Режим доступу: <https://www.postgresql.org/docs/16> (дата звернення: 05.06.2026).
5. TypeORM Documentation [Електронний ресурс] – Режим доступу: <https://typeorm.io/> (дата звернення: 05.06.2026).
6. Docker Documentation [Електронний ресурс] – Режим доступу: <https://docs.docker.com/> (дата звернення: 05.06.2026).
7. WayForPay Developer Guide [Електронний ресурс] – Режим доступу: <https://wiki.wayforpay.com/> (дата звернення: 05.06.2026).
8. OWASP Top Ten 2025 [Електронний ресурс] – Режим доступу: <https://owasp.org/www-project-top-ten/> (дата звернення: 05.06.2026).
9. Winters T. Software Engineering at Google: Lessons Learned from Programming over Time / T. Winters, T. Manshreck, H. Wright. - O'Reilly Media, 2022. - 602 с.
10. Geewax J. API Design Patterns / J. Geewax. - Manning Publications, 2022. - 600 с.
11. Casciaro M. Node.js Design Patterns: Design and implement production-grade Node.js applications / M. Casciaro, L. Mammino. - 3rd ed. - Packt Publishing, 2022. - 660 с.
12. Newman S. Building Microservices: Designing Fine-Grained Systems / S. Newman. - 2nd ed. - O'Reilly Media, 2022. - 640 с.
13. JWT.io Introduction [Електронний ресурс] – Режим доступу: <https://jwt.io/introduction> (дата звернення: 05.06.2026).

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 88   |

14. MDN Web Docs: CSS Grid Layout [Електронний ресурс] – Режим доступу: [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_grid\\_layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout) (дата звернення: 05.06.2026).
15. Google Lighthouse Documentation [Електронний ресурс] – Режим доступу: <https://developer.chrome.com/docs/lighthouse/> (дата звернення: 05.06.2026).
16. Zawacki-Richter O. Handbook of Open, Distance and Digital Education / O. Zawacki-Richter, I. Jung. - Springer, 2023. - 1183 с.
17. Chaffey D. Digital Business and E-Commerce Management / D. Chaffey. - 8th ed. - Pearson, 2022. - 680 с.
18. MDN Web Docs: Web Accessibility [Електронний ресурс] – Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/Accessibility> (дата звернення: 05.06.2026).
19. Render.com Documentation [Електронний ресурс] – Режим доступу: <https://render.com/docs> (дата звернення: 05.06.2026).
20. Twelve-Factor App Methodology [Електронний ресурс] – Режим доступу: <https://12factor.net/> (дата звернення: 05.06.2026).
21. Node.js Documentation [Електронний ресурс] – Режим доступу: <https://nodejs.org/en/docs> (дата звернення: 05.06.2026).
22. OpenAPI Specification 3.0 [Електронний ресурс] – Режим доступу: <https://spec.openapis.org/oas/v3.0.0> (дата звернення: 05.06.2026).
23. GitHub Actions Documentation [Електронний ресурс] – Режим доступу: <https://docs.github.com/en/actions> (дата звернення: 05.06.2026).
24. Interaction Design Foundation. The Encyclopedia of Human-Computer Interaction / Interaction Design Foundation. - 3rd ed. - IDF, 2023. - 520 с.
25. Mozilla Foundation. HTTP Overview [Електронний ресурс] – Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview> (дата звернення: 05.06.2026).
26. Passport.js Documentation [Електронний ресурс] – Режим доступу: <https://www.passportjs.org/docs/> (дата звернення: 05.06.2026).

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 89   |

27. bcrypt npm package [Електронний ресурс] – Режим доступу: <https://www.npmjs.com/package/bcrypt> (дата звернення: 05.06.2026).
28. WCAG 2.1 Guidelines [Електронний ресурс] – Режим доступу: <https://www.w3.org/TR/WCAG21/> (дата звернення: 05.06.2026).
29. Vernon V. Strategic Monoliths and Microservices: Driving Innovation Using Purposeful Architecture / V. Vernon, T. Jaskula. - Addison-Wesley, 2022. - 368 с.
30. Shvets A. Dive Into Design Patterns / A. Shvets. - Refactoring.Guru, 2022. - 406 с.
31. Rippon C. Learn React with TypeScript / C. Rippon. - 3rd ed. - Packt Publishing, 2025. - 444 с.
32. Colantonio J. Foundations of Modern Application Testing / J. Colantonio. - Test Guild LLC, 2023. - 298 с.
33. Kim G. The Unicorn Project: A Novel about Developers, Digital Disruption, and Thriving in the Age of Data / G. Kim. - IT Revolution Press, 2022. - 352 с.
34. Bass L. Software Architecture in Practice / L. Bass, P. Clements, R. Kazman. - 4th ed. - Addison-Wesley, 2022. - 448 с.

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 90   |

# ДОДАТКИ

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 91   |

```

auth.service.ts
import { Injectable, UnauthorizedException, ConflictException,
ForbiddenException, BadRequestException, NotFoundException } from
'@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { JwtService } from '@nestjs/jwt';
import { ConfigService } from '@nestjs/config';
import * as bcrypt from 'bcryptjs';
import * as crypto from 'crypto';
import { User } from '../users/user.entity';
import { RegisterDto, LoginDto, ForgotPasswordDto, ResetPasswordDto } from
'./auth.dto';
import { NotificationService } from '../notifications/notification.service';
import { ReferralService } from '../referral/referral.service';
import { EmailService } from './email.service';
import { fireAndForget } from '../common/logger.util';

```

```

@Injectable()
export class AuthService {
  constructor(
    @InjectRepository(User) private readonly userRepo: Repository<User>,
    private readonly jwtService: JwtService,
    private readonly config: ConfigService,
    private readonly notifSvc: NotificationService,
    private readonly referralSvc: ReferralService,
    private readonly emailSvc: EmailService,
  ) {}

  async register(dto: RegisterDto) {
    const exists = await this.userRepo.findOne({ where: { email: dto.email } });
    if (exists) throw new ConflictException('Користувач з таким email вже існує');
    const password = await bcrypt.hash(dto.password, 10);
    const user = this.userRepo.create({ ...dto, password });
    await this.userRepo.save(user);

    if (dto.referralToken) {
      const referrerId = this.referralSvc.decodeToken(dto.referralToken);
      if (referrerId && referrerId !== user.id) {
        fireAndForget(this.referralSvc.track(referrerId, user.id),
        'register:referralTrack');
      }
    }
  }
}

```

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 92   |

```

    fireAndForget(this.notifSvc.notifyAdminsNewUser(user.name, user.email,
user.id), 'register:notifyAdmins');
    return this.buildResponse(user);
}

async login(dto: LoginDto) {
    const user = await this.userRepo.findOne({ where: { email: dto.email } });
    if (!user) throw new UnauthorizedException('Невірний email або пароль');

    this.assertNotBanned(user);

    const ok = await bcrypt.compare(dto.password, user.password);
    if (!ok) throw new UnauthorizedException('Невірний email або пароль');
    return this.buildResponse(user);
}

async refresh(refreshToken: string) {
    try {
        const payload = this.jwtService.verify(refreshToken, {
            secret: this.config.get('JWT_REFRESH_SECRET'),
        });
        const user = await this.userRepo.findOne({ where: { id: payload.sub } });
        if (!user) throw new UnauthorizedException();

        this.assertNotBanned(user);

        return { accessToken: this.accessToken(user) };
    } catch (err) {
        if (err instanceof ForbiddenException) throw err;
        throw new UnauthorizedException('Refresh token недійсний');
    }
}

async googleLogin(profile: { googleId: string; name: string; email: string;
avatarUrl: string | null }) {
    let user = await this.userRepo.findOne({ where: { googleId: profile.googleId }
});

    if (!user) {
        user = await this.userRepo.findOne({ where: { email: profile.email } });
        if (user) {
            user.googleId = profile.googleId;
            if (!user.avatarUrl && profile.avatarUrl) user.avatarUrl = profile.avatarUrl;
            await this.userRepo.save(user);
        } else {

```

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 93   |

```

    user = this.userRepo.create({
      googleId: profile.googleId,
      email:    profile.email,
      name:     profile.name,
      avatarUrl: profile.avatarUrl,
      password: null as any,
    });
    await this.userRepo.save(user);
    fireAndForget(this.notifSvc.notifyAdminsNewUser(user.name, user.email,
user.id), 'googleLogin:notifyAdmins');
  }
}

this.assertNotBanned(user);
return this.buildResponse(user);
}

async forgotPassword(dto: ForgotPasswordDto): Promise<void> {
  const user = await this.userRepo.findOne({ where: { email: dto.email } });
  if (!user) return;

  const token = crypto.randomBytes(32).toString('hex');
  const hash  = crypto.createHash('sha256').update(token).digest('hex');

  user.resetPasswordToken = hash;
  user.resetPasswordExpires = new Date(Date.now() + 60 * 60 * 1000);
  await this.userRepo.save(user);

  const frontendUrl = this.config.get<string>('FRONTEND_URL',
'http://localhost:3001');
  const resetUrl = `${frontendUrl}/reset-password?token=${token}`;
  fireAndForget(this.emailSvc.sendPasswordReset(user.email, user.name,
resetUrl), 'forgotPassword:sendEmail');
}

async resetPassword(dto: ResetPasswordDto): Promise<void> {
  const hash = crypto.createHash('sha256').update(dto.token).digest('hex');
  const user = await this.userRepo.findOne({ where: { resetPasswordToken: hash
} });

  if (!user || !user.resetPasswordExpires || user.resetPasswordExpires < new
Date()) {
    throw new BadRequestException('Токен недійсний або прострочений');
  }
}

```

```

    user.password          = await bcrypt.hash(dto.password, 10);
    user.resetPasswordToken = null as any;
    user.resetPasswordExpires = null as any;
    await this.userRepo.save(user);
  }

  async findById(id: string) {
    return this.userRepo.findOne({ where: { id } });
  }

  assertNotBanned(user: User): void {
    if (!user.isActive) {
      const reason = user.banReason ?? 'Ваш акаунт заблоковано';
      throw new ForbiddenException(`Акаунт заблоковано: ${reason}`);
    }
  }

  private buildResponse(user: User) {
    return {
      accessToken: this.accessToken(user),
      refreshToken: this.refreshToken(user),
      user: {
        id: user.id, name: user.name, email: user.email,
        role: user.role, avatarUrl: user.avatarUrl ?? null,
      },
    };
  }

  private accessToken(user: User) {
    return this.jwtService.sign(
      { sub: user.id, email: user.email, role: user.role },
      { secret: this.config.get('JWT_SECRET'), expiresIn: '15m' },
    );
  }

  private refreshToken(user: User) {
    return this.jwtService.sign(
      { sub: user.id },
      { secret: this.config.get('JWT_REFRESH_SECRET'), expiresIn: '7d' },
    );
  }
}

```

certificate.service.ts

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 95   |

```
import { Injectable, NotFoundException, ConflictException, BadRequestException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { randomBytes } from 'node:crypto';
import * as PDFDocument from 'pdfkit';
import { Certificate } from './certificate.entity';
import { Course, Enrollment, Progress } from '../courses/course.entity';
import { User } from '../users/user.entity';
import { NotificationService } from '../notifications/notification.service';
import { fireAndForget } from '../common/logger.util';
```

@Injectable()

```
export class CertificateService {
```

```
  constructor(
```

```
    @InjectRepository(Certificate) private certRepo: Repository<Certificate>,
```

```
    @InjectRepository(Course) private courseRepo: Repository<Course>,
```

```
    @InjectRepository(Enrollment) private enrollmentRepo:
```

```
Repository<Enrollment>,
```

```
    @InjectRepository(Progress) private progressRepo: Repository<Progress>,
```

```
    private readonly notifSvc: NotificationService,
```

```
  ) {}
```

```
  async issue(courseId: string, user: User): Promise<Certificate> {
    const enrolled = await this.enrollmentRepo.findOne({ where: { userId: user.id,
courseId } });
    if (!enrolled) throw new BadRequestException('Ти не записаний на цей курс');
    const existing = await this.certRepo.findOne({ where: { userId: user.id, courseId
} });
    if (existing) throw new ConflictException('Сертифікат вже виданий');
    const course = await this.courseRepo.findOne({
      where: { id: courseId },
      relations: ['modules', 'modules.lessons', 'author'],
    });
    if (!course) throw new NotFoundException('Курс не знайдено');
    const allIds = course.modules.flatMap(m => m.lessons.map(l => l.id));
    if (!allIds.length) throw new BadRequestException('Курс не має уроків');
    const done = await this.progressRepo.createQueryBuilder('p')
      .where('p.user_id = :uid', { uid: user.id })
      .andWhere('p.lesson_id IN (:...ids)', { ids: allIds })
      .andWhere('p.completed = true').getCount();
    if (done < allIds.length)
      throw new BadRequestException(`Завершено ${done} з ${allIds.length}
уроків. Потрібно 100%`);
    const verifyCode = randomBytes(12).toString('hex').toUpperCase();
```

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 96   |

```

const pdfBuffer = await this.generatePdf({
  studentName: user.name,
  courseName: course.title,
  authorName: (course.author as any)?.name ?? 'Викладач',
  verifyCode,
  issuedAt: new Date(),
});
const pdfData = pdfBuffer.toString('base64');
const cert = await this.certRepo.save(
  this.certRepo.create({ userId: user.id, courseId, verifyCode, pdfData }),
);
fireAndForget(
  this.notifSvc.notifyCertificateIssued(user.id, courseId, course.title,
  verifyCode, null),
  'notif:notifyCertificateIssued',
);
return cert;
}

async verify(code: string) {
  const cert = await this.certRepo.findOne({
    where: { verifyCode: code },
    relations: ['user', 'course'],
  });
  if (!cert) throw new NotFoundException('Сертифікат не знайдено');
  return {
    valid: true,
    studentName: cert.user.name,
    courseName: cert.course.title,
    issuedAt: cert.issuedAt,
    verifyCode: cert.verifyCode,
  };
}

async getPdfBuffer(code: string, userId: string): Promise<Buffer> {
  const cert = await this.certRepo.findOne({ where: { verifyCode: code, userId }
});
  if (!cert || !cert.pdfData) throw new NotFoundException('Сертифікат не
знайдено');
  return Buffer.from(cert.pdfData, 'base64');
}

private generatePdf(data: {
  studentName: string; courseName: string;
  authorName: string; verifyCode: string; issuedAt: Date;

```

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 97   |

```

}): Promise<Buffer> {
  return new Promise((resolve, reject) => {
    const path = require('path');
    const fontsDir = path.join(__dirname, '..', 'certificates', 'fonts');
    const regular = path.join(fontsDir, 'Roboto-Regular.ttf');
    const bold = path.join(fontsDir, 'Roboto-Bold.ttf');
    const doc = new PDFDocument({ size: 'A4', layout: 'landscape', margin: 0,
autoFirstPage: true });
    const chunks: Buffer[] = [];
    doc.on('data', c => chunks.push(c));
    doc.on('end', () => resolve(Buffer.concat(chunks)));
    doc.on('error', reject);
    doc.registerFont('Regular', regular);
    doc.registerFont('Bold', bold);
    const W = doc.page.width;
    const H = doc.page.height;
    doc.rect(0, 0, W, H).fill('#fafaf8');
    doc.rect(20, 20, W - 40, H - 40).lineWidth(2).stroke('#c9a84c');
    doc.fontSize(40).fillColor('#1e1b4b').font('Bold')
      .text('СЕРТИФІКАТ', 0, 70, { align: 'center', width: W });
    doc.fontSize(13).fillColor('#6b7280').font('Regular')
      .text('про успішне завершення курсу', 0, 122, { align: 'center', width: W });
    doc.moveTo(W * 0.2, 152).lineTo(W * 0.8,
152).lineWidth(1).stroke('#c9a84c');
    doc.fontSize(28).fillColor('#111827').font('Bold')
      .text(data.studentName, 60, 168, { align: 'center', width: W - 120 });
    doc.fontSize(19).fillColor('#4f46e5').font('Bold')
      .text(`"${data.courseName}"`, 80, 238, { align: 'center', width: W - 160 });
    const d = data.issuedAt;
    const dateStr =
`"${d.getDate().toString().padStart(2, '0')}.${(d.getMonth()+1).toString().padStart(2,
'0')}.${d.getFullYear()}`;
    doc.fontSize(11).fillColor('#6b7280').font('Regular')
      .text(`Дата видачі: ${dateStr}`, 60, 360);
    doc.fontSize(11).fillColor('#6b7280').font('Regular')
      .text(`Викладач: ${data.authorName}`, W / 2, 360);
    doc.fontSize(9).fillColor('#9ca3af').font('Regular')
      .text(`Код верифікації: ${data.verifyCode}`, 0, 400, { align: 'center', width:
W });
    doc.end();
  });
}

```

wayforpay.service.ts

```
import { Injectable, BadRequestException } from '@nestjs/common';
import { ConfigService } from '@nestjs/config';
import * as crypto from 'node:crypto';
```

```
@Injectable()
```

```
export class WayForPayService {
```

```
  private readonly merchantAccount: string;
  private readonly merchantSecretKey: string;
  private readonly merchantDomain: string;
  private readonly resultUrl: string;
  private readonly serverUrl: string;
```

```
  constructor(private readonly config: ConfigService) {
    this.merchantAccount
config.getOrThrow('WFP_MERCHANT_ACCOUNT');
    this.merchantSecretKey
config.getOrThrow('WFP_MERCHANT_SECRET_KEY');
    this.merchantDomain
config.getOrThrow('WFP_MERCHANT_DOMAIN');
    this.resultUrl      = config.getOrThrow('WFP_RESULT_URL');
    this.serverUrl      = config.getOrThrow('WFP_SERVER_URL');
  }
```

```
  createPaymentForm(params: {
    orderId: string; amount: number;
    description: string; courseId: string; userId: string;
  }): { formData: Record<string, string>; action: string } {
    const orderDate = Math.floor(Date.now() / 1000);
    const productName = params.description;
    const productCount = '1';
    const productPrice = String(Number(params.amount));
    const amount = Number(params.amount);
    const currency = 'UAH';
    const signatureString = [
      this.merchantAccount, this.merchantDomain, params.orderId,
      orderDate, amount, currency, productName, productCount, productPrice,
    ].join(';');
    const merchantSignature = this.sign(signatureString);
    const formData: Record<string, string> = {
      merchantAccount: this.merchantAccount,
      merchantAuthType: 'SimpleSignature',
      merchantDomainName: this.merchantDomain,
      merchantSignature,
```

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 99   |

```

        orderReference: params.orderId,
        orderDate:      String(orderDate),
        amount:         String(amount),
        currency,
        orderTimeout:   '49000',
        'productName[]': productName,
        'productCount[]': productCount,
        'productPrice[]': productPrice,
        returnUrl:      this.resultUrl,
        serviceUrl:     this.serverUrl,
    };
    return { formData, action: 'https://secure.wayforpay.com/pay' };
}

verifyCallback(body: Record<string, any>): {
    valid: boolean; orderId: string; status: string;
    amount: number; type: 'course' | 'subscription';
    courseId?: string; userId: string;
} {
    const {
        merchantAccount, orderReference, amount, currency,
        authCode, cardPan, transactionStatus, reasonCode,
    } = body;
    const signatureString = [
        merchantAccount, orderReference, amount, currency,
        authCode ?? "", cardPan ?? "", transactionStatus ?? "", reasonCode ?? "",
    ].join(';');
    const expectedSignature = this.sign(signatureString);
    if (body.merchantSignature !== expectedSignature) {
        throw new BadRequestException('Невірний підпис WayForPay');
    }
    const parts = orderReference.split('_');
    const isSubscription = parts[0] === 'sub';
    if (isSubscription) {
        return {
            valid: true,
            orderId: orderReference,
            status: transactionStatus === 'Approved' ? 'success' : 'failure',
            amount: Number(amount),
            type: 'subscription',
            userId: parts[parts.length - 2],
            plan: parts[1],
        } as any;
    }
}

const toUUID = (hex: string) =>

```

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 100  |

```

        `${hex.slice(0,8)}-${hex.slice(8,12)}-${hex.slice(12,16)}-${hex.slice(16,20)}-${hex.slice(20)}`;
        return {
            valid: true,
            orderId: orderReference,
            status: transactionStatus === 'Approved' ? 'success' : 'failure',
            amount: Number(amount),
            type: 'course',
            courseId: toUUID(parts[parts.length - 3]),
            userId: toUUID(parts[parts.length - 2]),
        };
    }

    buildCallbackResponse(orderReference: string, status: 'accept' | 'decline') {
        const time = Math.floor(Date.now() / 1000);
        const signatureString = [orderReference, status, time].join(';');
        return { orderReference, status, time, signature: this.sign(signatureString) };
    }

    private sign(data: string): string {
        return crypto.createHmac('md5',
            this.merchantSecretKey).update(data).digest('hex');
    }
}

```

|      |      |                  |        |      |                          |      |
|------|------|------------------|--------|------|--------------------------|------|
|      |      | Головня С.М.     |        |      | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
|      |      | Лобанчикова Н.М. |        |      |                          |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                          | 101  |