

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОЇ
ОСВІТНЬОЇ ПЛАТФОРМИ
DEVELOPMENT OF AN INTERACTIVE
LEARNING PLATFORM

Виконав(ла): студент(ка) 4 курсу, групи КНТ-112

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПАСТУШЕНКО М.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ДЬЯЧУК Т.С.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПАСТУШЕНКА Микити Олександровича
 (ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація інтерактивної освітньої платформи. Development of an Interactive Learning Platform
 керівник проєкту (роботи) к.т.н., доцент ГОФМАН Євгеній Олександрович,
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)
 затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 10 червня 2026 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області та постановка завдання дослідження. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми.
5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГОФМАН Є.О., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання «20» квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ **Микита ПАСТУШЕНКО**
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ **Євгеній ГОФМАН**
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 95 с., 4 табл., 23 рис., 3 дод., 19 джерел.

ОСВІТНЯ ПЛАТФОРМА, ГЕЙМІФІКАЦІЯ, ВЕБЗАСТОСУНОК, ІГРОВІ МЕХАНІКИ, СИСТЕМА ДОСЯГНЕНЬ, ДИСТАНЦІЙНЕ НАВЧАННЯ, ТРЕКІНГ ПРОГРЕСУ.

Об'єкт дослідження – процес інженерії програмного забезпечення для дистанційного навчання з використанням мотиваційних механік.

Предмет дослідження – методи та програмні інструменти для побудови освітніх вебплатформ з інтегрованою системою гейміфікації.

Мета роботи – підвищення рівня залученості користувачів та зниження трудомісткості адміністрування навчального процесу за рахунок програмної реалізації інтерактивної вебплатформи з автоматизованим відстеженням прогресу, системою квестів та ігрових досягнень.

Матеріали, методи та технічні засоби: використано методи компонентно-орієнтованого програмування та Server-Side Rendering (SSR). Серверну та клієнтську частини реалізовано на базі фреймворку Next.js 16 (App Router) із використанням React 19 та мови TypeScript. Інтерфейс побудовано за допомогою утилітарного CSS-фреймворку Tailwind CSS. Для взаємодії з реляційною базою даних PostgreSQL використано Prisma ORM (v7) з офіційним PG-адаптером. Безпеку платформи реалізовано через JWT-токени (бібліотека jose), HTTP-only cookies та криптографічне хешування паролів за допомогою bcryptjs. Розроблення проводилося в середовищі Visual Studio Code на персональному комп'ютері.

Результати. Розроблено повнофункціональний вебзастосунок «EduSteam», який забезпечує розподіл користувачів за ролями (Студент, Викладач), управління профілями, створення та проходження навчальних

курсів. Реалізовано систему гейміфікації, яка автоматично відстежує витрачений на навчання час, нараховує очки досвіду (XP) за виконання квестів та видає глобальні й локальні досягнення (ачівки). Впроваджено персональні дашборди користувачів, налаштування приватності та безпечну транзакційну реєстрацію.

Висновки. Досягнуто мети роботи: створено освітній інструмент, який автоматизує навчальний процес та стимулює користувачів до безперервної освіти завдяки ігровим елементам. Застосунок забезпечує високий рівень безпеки та цілісності даних при роботі з курсами та прогресом студентів, підтримує сучасний адаптивний дизайн.

Галузь використання – заклади вищої освіти, приватні онлайн-школи, курси підвищення кваліфікації та платформи для корпоративного навчання.

Економічна ефективність. Використання застосунку знижує відсоток незавершених курсів (зменшує відтік студентів), автоматизує контроль успішності, звільняючи час викладачів, та підвищує загальну рентабельність освітнього закладу за рахунок утримання уваги користувачів на платформі.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor:
95 pages, 4 tables, 23 figures, 3 appendixes, 19 sources.

EDUCATIONAL PLATFORM, GAMIFICATION, WEB APPLICATION, GAME MECHANICS, ACHIEVEMENT SYSTEM, DISTANCE LEARNING, PROGRESS TRACKING.

Object of research: the software engineering process for distance learning using motivational mechanics.

Subject of research: methods and software tools for building educational web platforms with an integrated gamification system.

The purpose of the work is to increase the level of involvement and motivation of learners by creating an interactive platform with progress tracking, a quest system, and game achievements.

Materials, methods, and tools: component-oriented programming methods and Server-Side Rendering (SSR) were used. The server and client parts are implemented based on the Next.js 16 framework (App Router) using React 19 and TypeScript. The interface is built using the Tailwind CSS utility-first framework. To interact with the PostgreSQL relational database, Prisma ORM (v7) with the official PG adapter was used. Platform security is implemented through JWT tokens (jose library), HTTP-only cookies, and cryptographic password hashing using bcryptjs. The development was carried out in the Visual Studio Code environment on a personal computer.

Results. A fully functional web application "EduSteam" has been developed, which provides user role distribution (Student, Teacher), profile management, and the creation and completion of educational courses. A gamification system has been implemented that automatically tracks the time spent on learning, awards experience points (XP) for completing quests, and

grants global and local achievements. Personal user dashboards, privacy settings, and secure transactional registration have been introduced.

Conclusions. The goal of the work has been achieved: an educational tool has been created that automates the learning process and stimulates users to continuous education through game elements. The application ensures a high level of security and data integrity when working with courses and student progress, and supports a modern responsive design.

Field of use is higher education institutions, private online schools, professional development courses, and platforms for corporate training.

Economic efficiency. The use of the application reduces the percentage of incomplete courses (decreases student dropout rates), automates performance monitoring, freeing up teachers' time, and increases the overall profitability of the educational institution by retaining user attention on the platform.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	10
Вступ.....	11
1 Аналіз предметної області та постановка завдання дослідження.....	14
1.1 Короткий опис досліджуваної предметної області	14
1.2 Огляд існуючих рішень проблеми	16
1.3 Опис основних вимог до програми.....	19
1.4 Постановка завдання роботи	23
1.5 Висновки за розділом 1	23
2 Розробка архітектури програми.....	25
2.1 Основні компоненти системи.....	25
2.2 Функціонування системи	26
2.3 Вибір мови програмування та інструментарію	27
2.4 Вимоги до технічних і програмних засобів	30
2.5 Висновки за розділом 2	30
3 Розробка програми	32
3.1 Архітектура системи	32
3.2 Функціонування системи	34
3.3 Проектування бази даних.....	35
3.4 Висновки за розділом 3	40
4 Експлуатація, тестування та експериментальне дослідження програми .	41
4.1 Опис застосування програми.....	41
4.2 Інструкція по роботі з програмою	42
4.3 Приклад роботи користувача з програмою.....	49
4.4 Тестування роботи програми.....	53
4.5 Рекомендації щодо використання	59
4.6 Плани з розвитку.....	59

Висновки	61
Перелік джерел посилання	64
Додаток А Технічне завдання	66
Додаток Б Текст програми	73
Додаток В Слайди презентації.....	88

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ACID	– Atomicity, Consistency, Isolation, Durability;
API	– Application Programming Interface;
CSS	– Cascading Style Sheets;
DB	– Database;
DBMS	– Database Management System;
FCP	– First Contentful Paint;
HTML	– HyperText Markup Language;
HTTP	– HyperText Transfer Protocol;
JSON	– JavaScript Object Notation;
JWT	– JSON Web Token;
LMS	– Learning Management System;
MOOC	– Massive Open Online Courses;
ORM	– Object-Relational Mapping;
REST	– Representational State Transfer;
RSC	– React Server Components;
SDT	– Self-Determination Theory;
SSR	– Server-Side Rendering;
TS	– TypeScript;
TTI	– Time to Interactive;
UI	– User Interface;
UX	– User Experience;
XP	– Experience Points;
XSS	– Cross-Site Scripting;
БД	– база даних;
СУБД	– система управління базами даних.

ВСТУП

Сучасний етап розвитку інформаційного суспільства супроводжується докорінною трансформацією освітніх парадигм. Сьогодні визначальною умовою успішного дистанційного навчання є не лише доступ до якісного цифрового контенту, а й здатність програмного середовища ефективно утримувати увагу слухачів. Незважаючи на широке розповсюдження різноманітних навчальних систем, значна частина освітніх закладів стикається з критичною проблемою зниження мотивації студентів. Саме тому розробка інтерактивних платформ, здатних органічно поєднувати академічну подачу матеріалу із мотиваційними ігровими елементами, набуває особливої наукової та практичної значущості.

Аналіз світового ринку освітніх технологій (EdTech) свідчить, що виключно пасивне споживання інформації призводить до різкого падіння результативності навчання. Дослідження у сфері масових відкритих онлайн-курсів (МООС) підтверджують високий рівень відсіву користувачів саме через брак інтерактивної взаємодії та нестачу механізмів заохочення. Хоча традиційні системи управління навчанням (LMS) і забезпечують базовий адміністративний функціонал, вони часто виявляються надто консервативними та негнучкими у питаннях утримання аудиторії. Відповідно, виникає гостра потреба в інноваційних програмних рішеннях, які б дозволили легко інтегрувати методи гейміфікації в повсякденний освітній процес.

Відтак, фундаментальна проблема полягає у відсутності спеціалізованого, швидкодіючого вебзастосунку, який би пропонував викладачам зручний інструментарій для публікації матеріалів, а студентам – персоналізований ігровий досвід із автоматичним трекінгом успішності. Огляд популярних аналогів (наприклад, Moodle чи Coursera) виявляє їхню архітектурну перевантаженість або обмеженість у налаштуванні ігрових механік під конкретні цілі. Водночас сучасні можливості веброзробки,

зокрема використання фреймворку Next.js та екосистеми React, відкривають широкі перспективи для створення високопродуктивних платформ на базі серверного рендерингу (SSR) та надійно типізованого коду [1]-[2].

Об'єктом дослідження є процес організації та автоматизації дистанційного навчання з використанням мотиваційних ігрових механік.

Предметом дослідження є методи, інструменти та програмні засоби побудови клієнт-серверних освітніх вебплатформ із вбудованими функціями гейміфікації.

Мета роботи полягає у проєктуванні та програмній реалізації комплексної вебекосистеми «EduSteam», спрямованої на підвищення ефективності навчання шляхом інтеграції інтерактивних завдань, системи досягнень та автоматизованого відстеження прогресу користувачів.

Для досягнення поставленої мети було визначено наступні завдання дослідження:

- здійснити системний аналіз принципів гейміфікації в освіті та дослідити недоліки наявних LMS-систем для формування архітектурних вимог;
- спроектувати реляційну модель бази даних та клієнт-серверне середовище, орієнтоване на швидку обробку запитів та забезпечення безпеки даних;
- розробити функціональні модулі платформи: управління профілями, створення курсів, трекінг навчального часу та алгоритми нарахування очок досвіду (XP);
- виконати комплексне тестування розробленого програмного забезпечення для перевірки його стабільності, швидкодії та коректності роботи рольової моделі (Студент/Викладач).

Методологічну базу дослідження формують концепції компонентно-орієнтованої розробки та сучасні патерни проєктування вебзастосунків. Технічну основу реалізації складають фреймворк Next.js

для створення інтерфейсу та обробки серверної логіки, мова програмування TypeScript, об'єктно-реляційне відображення Prisma ORM, а також система управління базами даних PostgreSQL [1]-[5].

Практичне значення отриманих результатів. Створений програмний продукт становить безпосередню цінність для вищих навчальних закладів, платформ підвищення кваліфікації та незалежних менторів, оскільки дозволяє суттєво зменшити рівень відсіву студентів. Запровадження інструментів розробленої платформи сприяє прозорості навчального процесу та стимулює систематичне засвоєння матеріалу завдяки грамотному застосуванню ігрової мотивації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

1.1 Короткий опис досліджуваної предметної області

Сучасний етап розвитку інформаційних технологій характеризується глобальною цифровізацією всіх сфер суспільного життя, серед яких освітній сектор займає одне з провідних місць. Дистанційна освіта (e-learning) пройшла шлях від експериментальних онлайн-курсів до повноцінної альтернативи традиційному академічному навчанню. Її популярність зумовлена низкою беззаперечних переваг: гнучкістю графіка, відсутністю географічних бар'єрів, можливістю індивідуалізації темпу навчання та широким доступом до мультимедійного контенту.

Проте, незважаючи на стрімкий розвиток платформ масових відкритих онлайн-курсів (МООС) та систем управління навчанням (LMS), галузь дистанційної освіти стикається з фундаментальною проблемою – критично низьким рівнем утримання студентів (Retention Rate). Згідно з дослідженнями аналітичних центрів, лише від 5% до 15% користувачів, які реєструються на безкоштовні або базові онлайн-курси, повністю завершують їх. Основною причиною такої негативної статистики є специфіка психології дистанційного навчання: відсутність прямого соціального контролю, відчуття ізольованості студента та швидке згасання початкової мотивації через монотонність процесу споживання інформації.

Ефективним інструментом вирішення проблеми низької залученості є гейміфікація (ігрофікація) – процес інтеграції ігрових механік та принципів ігрового дизайну в неігровий (у даному випадку освітній) контекст. Гейміфікація базується на глибокому розумінні поведінкової психології. Вона стимулює як зовнішню мотивацію (через отримання нагород), так і внутрішню (через відчуття власного прогресу, компетентності та досягнення мети) [6]-[7].

У контексті освітніх вебплатформ гейміфікація найчастіше реалізується через наступні ключові механіки:

- очки досвіду (Experience Points, XP). Базова кількісна метрика, що відображає зусилля студента. Нарахування XP за виконання завдань, проходження тестів або просто за час, проведений за навчанням, створює миттєвий зворотний зв'язок (feedback loop), підтверджуючи цінність виконаної дії [7];

- рівні (Levels). Система ієрархії, що базується на накопичених очках досвіду. Перехід на новий рівень слугує довгостроковим мотиватором та візуалізує загальний прогрес користувача на платформі;

- досягнення та відзнаки (Achievements / Badges). Віртуальні нагороди за виконання специфічних умов або подолання певних викликів (наприклад, "Навчався 7 днів поспіль" або "Завершив перший курс на відмінно"). Це задовольняє психологічну потребу в колекціонуванні та визнанні [6];

- квести та місії. Перетворення стандартних навчальних модулів на сюжетні або цільові завдання з чітко визначеною винагородою, що додає процесу навчання елемент інтриги та дослідження [8].

Синтез технологій дистанційної освіти та гейміфікації формує нову предметну область – інтерактивні освітні екосистеми (EdTech). Такі системи вимагають не лише продумування педагогічного дизайну, але й використання сучасних програмних архітектур, здатних у реальному часі обробляти дані про прогрес користувачів, безпечно зберігати інформацію про їхні досягнення та миттєво оновлювати клієнтський інтерфейс.

Отже, досліджувана предметна область охоплює методи проєктування та розробки програмного забезпечення, що здатне надійно поєднати класичне управління навчальним контентом (робота викладачів) із сучасними мотиваційними ігровими алгоритмами (робота студентів) [7].

1.2 Огляд існуючих рішень проблеми

Для формування об'єктивних вимог до розроблюваної системи необхідно провести ґрунтовний аналіз наявних на ринку програмних рішень. Ринок освітніх технологій (EdTech) пропонує широкий спектр продуктів, які можна умовно розділити на три категорії: масові відкриті онлайн-курси (МООС), вузькоспеціалізовані гейміфіковані додатки та класичні академічні системи управління навчанням (LMS). Для детального розгляду було обрано найбільш репрезентативних представників з кожної категорії.

1.2.1 Освітня платформа Coursera

Coursera є одним із найбільших у світі провайдерів масових відкритих онлайн-курсів, що співпрацює з провідними університетами та корпораціями. Платформа пропонує високоякісний відеоконтент, систему взаємного оцінювання (peer-to-peer) та можливість отримання верифікованих сертифікатів [9].

Переваги:

- величезна база структурованих матеріалів від світових експертів;
- хмарна інфраструктура, здатна витримувати великі навантаження;
- зручний відеоплеєр із інтерактивними субтитрами [9].

Недоліки:

- практично повна відсутність механік гейміфікації. Система мотивації базується виключно на отриманні фінального сертифіката, що не працює для студентів, яким потрібна щоденна підтримка інтересу;
- високий відсоток відсіву (drop-out rate) на безкоштовних програмах через брак інтерактивності. Платформа фактично переносить застарілу лекційну модель у цифровий формат без адаптації під психологію сучасного користувача [9].

1.2.2 Гейміфікований сервіс Duolingo

Duolingo – це найпопулярніший у світі сервіс для вивчення іноземних мов, який є еталоном впровадження гейміфікації в освітній процес. Застосунок базується на концепції мікронавчання (bite-sized learning) [10].

Переваги:

- глибока інтеграція ігрових механік: серії безперервного навчання (streaks), очки досвіду, лідерборди, ліги та віртуальна валюта [10];
- неймовірно високий рівень утримання користувачів завдяки грамотному використанню push-сповіщень та миттєвому зворотному зв'язку;
- дружній, інтуїтивно зрозумілий UI/UX дизайн [10].

Недоліки:

- вузька предметна орієнтованість. Платформа не підходить для розміщення складних, багаторівневих курсів з програмування, інженерії чи точних наук;
- відсутність інструментарію для сторонніх викладачів. Користувач не може створити та опублікувати власний курс для своєї аудиторії – весь контент генерується та контролюється виключно розробниками платформи.

1.2.3 Академічна система управління навчанням Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment) – це безкоштовна LMS-система з відкритим вихідним кодом, яка є стандартом де-факто для більшості державних університетів та шкіл по всьому світу [11].

Переваги:

- відкритий вихідний код (Open Source), що дозволяє розгорнути систему на власних серверах;
- величезна кількість налаштувань для адміністраторів: управління ролями, розкладом, журналом оцінок та тестуванням;
- можливість розширення функціоналу за допомогою плагінів [11].

Недоліки:

- застарілий та перевантажений інтерфейс користувача (UI). Система створювалася розробниками для адміністраторів, а не для комфорту студентів;
- складність розгортання та підтримки гейміфікації. Щоб додати базові ігрові елементи (бали чи бейджі), необхідно встановлювати та налаштовувати сторонні плагіни, які часто конфліктують між собою та уповільнюють роботу сервера;
- важка клієнтська частина, яка повільно працює на мобільних пристроях без додаткової оптимізації [11].

1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів

На основі проведеного аналізу було сформовано порівняльну таблицю (табл. 1.1), яка демонструє ключові відмінності між існуючими аналогами та проєктованою платформою «EduSteam».

Аналіз конкурентного середовища показав, що на ринку існує незаповнена ніша. Сучасним викладачам та студентам потрібен інструмент, який би поєднував легкість створення навчального контенту (як у Moodle чи Udemu), але при цьому мав би глибоко інтегроване, "з коробки", ядро гейміфікації (як у Duolingo) та працював на сучасній, швидкій архітектурі. Саме ці принципи покладено в основу розробки вебзастосунку «EduSteam».

Таблиця 1.1 – Таблиця порівняння платформ

Критерій порівняння	Coursera	Duolingo	Moodle	EduSteam (Розробка)
Основний фокус платформи	Масові лекції	Мікронавчання	Адміністрування	Інтерактивні курси
Ступінь гейміфікації	Низький	Дуже високий	Базовий (через плагіни)	Високий (XP, досягнення)
Створення кастомних курсів викладачами	Лише для партнерів	Неможливо	Так (але складно)	Так (інтуїтивно)
Створення кастомних Автоматичний трекінг часу навчання	Частково	Ні	Так	Так (у реальному часі)
Архітектура клієнтської частини	SPA (React)	SPA / Native	Monolith (PHP)	SSR (Next.js)
Сучасний UI/UX дизайн	Добре	Відмінно	Незадовільно	Відмінно
Складність розгортання системи	Хмарна (SaaS)	Хмарна (SaaS)	Висока (Self-hosted)	Низька (Docker/Vercel)

1.3 Опис основних вимог до програми

На етапі проєктування системи критично важливим є чітке визначення вимог, яким повинен відповідати майбутній програмний продукт. Вимоги до освітньої вебплатформи «EduSteam» поділяються на дві основні категорії: функціональні (описують, що повинна робити система) та нефункціональні (описують, як система повинна працювати).

1.3.1 Визначення функціональних вимог

Функціональні вимоги формуються на основі рольової моделі системи. У розроблюваному застосунку передбачено дві основні групи авторизованих користувачів: «Студент» та «Викладач», а також базовий рівень доступу для неавторизованих відвідувачів («Гість»).

Функціональні вимоги для ролі «Гість»:

- перегляд головної сторінки платформи з описом її переваг;
- доступ до публічного каталогу курсів (без можливості перегляду закритого навчального контенту);
- реєстрація нового облікового запису в системі з вибором базової ролі;
- авторизація за допомогою електронної пошти та пароля.

Функціональні вимоги для ролі «Студент»:

- управління власним профілем: завантаження аватара, редагування біографії, налаштування приватності (видимість профілю для інших);
- запис на обрані курси з каталогу;
- перегляд навчальних матеріалів (текстові лекції, відеоматеріали);
- автоматичне відстеження часу, витраченого на проходження курсу (трекінг у хвилинах);
- виконання внутрішніх завдань (квестів) у межах курсу;
- отримання очок досвіду (XP) за успішне завершення квестів та накопичений час навчання;
- підвищення власного рівня (Level) на основі загальної кількості XP;
- здобуття глобальних та локальних ігрових досягнень (Achievements) за виконання певних умов;
- перегляд персонального дашборду з візуалізацією статистики прогресу.

Функціональні вимоги для ролі «Викладач»:

- усі базові можливості ролі «Студент»;
- створення нових навчальних курсів із зазначенням назви, опису, категорії та рівня складності;
- формування структури курсу (додавання модулів та уроків);
- створення квестів для студентів із налаштуванням кількості XP, яка видається за їх виконання;
- перегляд базової статистики щодо кількості студентів, зарахованих на створені курси;
- редагування або видалення власного навчального контенту.

Діаграму варіантів використання відображено на рисунку 1.1.



Рисунок 1.1 – Діаграма варіантів використання системи «EduSteam»

1.3.2 Нефункціональні вимоги

Нефункціональні вимоги визначають атрибути якості системи, її надійність, безпеку та продуктивність.

Вимоги до швидкодії та продуктивності:

- вебзастосунок має використовувати технологію Server-Side Rendering (SSR) для забезпечення швидкого початкового завантаження сторінок (First Contentful Paint);

- час відгуку сервера на стандартні запити до бази даних не повинен перевищувати 500 мс.

Вимоги до безпеки:

- паролі користувачів заборонено зберігати у відкритому вигляді, необхідно використовувати криптографічне хешування за алгоритмом bcrypt із застосуванням динамічної солі (salt);

- для управління сесіями повинна використовуватися технологія JSON Web Tokens (JWT), токени повинні зберігатися виключно у HTTP-only cookies для захисту від атак типу XSS (Cross-Site Scripting);

- усі маршрути API та сторінки, що вимагають авторизації, повинні бути захищені за допомогою проміжного програмного забезпечення (Middleware).

Вимоги до ергономіки та інтерфейсу (UX/UI):

- інтерфейс має бути адаптивним (Responsive Web Design), тобто коректно відображатися на десктопних комп'ютерах, планшетах та смартфонах;

- дизайн повинен відповідати сучасним тенденціям мінімалізму та не відволікати студента від навчального процесу.

Вимоги до бази даних:

- система має використовувати реляційну базу даних для гарантії цілісності інформації через зовнішні ключі (Foreign Keys) та підтримку транзакцій (ACID).

1.4 Постановка завдання роботи

Зважаючи на результати аналізу предметної області та сформовані вимоги, загальне завдання дипломної роботи полягає у створенні повністю працездатного клієнт-серверного вебзастосунку «EduSteam».

Для реалізації цього завдання необхідно виконати наступні етапи розробки:

- розробити структуру реляційної бази даних, яка об'єднуватиме логіку навчання (курси, записи) та логіку гейміфікації (XP, рівні, квести, досягнення);
- налаштувати серверну взаємодію за допомогою ORM-системи для безпечної обробки даних;
- реалізувати алгоритми реєстрації, авторизації та розмежування прав доступу користувачів;
- розробити клієнтський інтерфейс на базі компонентного підходу, реалізувавши інтерактивні панелі для викладачів та студентів;
- запрограмувати ядро гейміфікації: тригери нарахування балів, перерахунок рівнів та відстеження часу;
- провести тестування системи для виявлення та усунення потенційних вразливостей або архітектурних помилок.

1.5 Висновки за розділом 1

У першому розділі пояснювальної записки було проведено комплексний аналіз предметної області дистанційного навчання та гейміфікації. Обґрунтовано актуальність впровадження ігрових механік в освітній процес як дієвого засобу боротьби з низькою мотивацією студентів.

Здійснено огляд існуючих на ринку рішень (Coursera, Duolingo, Moodle), виявлено їхні переваги та недоліки, що дозволило знайти вільну

нішу для нового продукту. На основі аналізу було сформовано концепцію розроблюваної платформи «EduSteam». Визначено чіткі функціональні вимоги для кожної ролі користувача та ключові нефункціональні вимоги до архітектури, швидкодії та безпеки.

Встановлені у цьому розділі вимоги є фундаментом для наступного етапу розробки – проєктування архітектури системи та вибору конкретного стеку технологій, що буде детально розглянуто у другому розділі.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

Проектування архітектури вебплатформи «EduSteam» базується на принципах модульності та чіткого розмежування зон відповідальності. Обрана модель дозволяє забезпечити масштабованість системи та легкість її підтримки.

2.1 Основні компоненти системи

Архітектура застосунку побудована за трирівневою моделлю (Three-tier architecture), яка включає такі складові:

- рівень представлення (Presentation Layer / Frontend). Реалізовано за допомогою фреймворку Next.js та бібліотеки React. Відповідає за візуалізацію даних, взаємодію з користувачем та обробку подій на клієнтській стороні;
- рівень логіки (Logic Layer / Backend). У межах Next.js App Router цей рівень представлений серверними компонентами (RSC) та серверними функціями (Route Handlers). Тут зосереджена бізнес-логіка: аутентифікація, перевірка прав доступу, розрахунок ігрового досвіду (XP) та управління контентом курсів;
- рівень даних (Data Layer / Database). Включає реляційну базу даних PostgreSQL, де зберігається вся постійна інформація та шар доступу до даних у вигляді Prisma ORM.

2.1.1 Взаємодія компонентів системи

Взаємодія між компонентами відбувається за протоколом HTTP/HTTPS. Клієнтська частина надсилає запити до серверних маршрутів, які, своєю чергою, звертаються до бази даних через об'єктно-реляційне відображення. Особливістю архітектури є широке використання

Server-Side Rendering (SSR): сервер не просто віддає дані, а формує готову HTML-розмітку, що містить актуальну інформацію про прогрес студента, і надсилає її браузеру. Це мінімізує кількість запитів від клієнта після завантаження сторінки та підвищує загальну продуктивність [12].

2.2 Функціонування системи

2.2.1 Основні процеси системи

Функціонування платформи «EduSteam» забезпечується злагодженою роботою декількох ключових алгоритмів:

- процес ідентифікації та доступу. При вході в систему сервер генерує JWT-токен, який шифрується та записується в HTTP-only cookies. Кожен наступний запит користувача автоматично перевіряється проміжним ПЗ (Middleware) на предмет валідності токена та відповідності ролі користувача запитуваному ресурсу;

- процес накопичення досвіду та трекінгу часу. Система використовує механізм активного відстеження сесії. Під час вивчення курсу клієнтська частина фіксує активність користувача. При виконанні певних тригерів (завершення уроку або певний інтервал часу) надсилається запит на сервер, який через Prisma оновлює значення `total_minutes` та додає відповідну кількість XP у таблицю `User`;

- процес генерації досягнень. Після оновлення досвіду спрацьовує логіка перевірки умов досягнень. Якщо нові значення показників відповідають критеріям (наприклад, сумарно 1000 XP), у базі даних створюється запис про здобуття ачивки, що миттєво відображається в профілі студента.

2.2.2 Логіка роботи системи

Логіка системи побудована на декларативному підході. Стан інтерфейсу безпосередньо залежить від стану даних у базі. Завдяки використанню TypeScript, усі дані, що циркулюють між базою, сервером та клієнтом, мають сувору типізацію. Це виключає ситуації, коли компонент намагається відобразити неіснуюче поле або отримати дані неправильного типу (наприклад, рядок замість числа для рівня XP) [13].

2.3 Вибір мови програмування та інструментарію

2.3.1 Порівняльний аналіз та обґрунтування вибору мови програмування

Ефективність розробки освітньої платформи та її подальша стабільність безпосередньо залежать від обраної мови програмування. У сучасній веброзробці основними конкурентами для створення серверної та клієнтської логіки є JavaScript, Python та TypeScript. Проведемо їх порівняльну характеристику (табл. 2.1) [3].

Таблиця 2.1 – Порівняльна характеристика мов програмування

Критерій порівняння	JavaScript	Python	TypeScript
Типізація	Динамічна	Динамічна	Суворо статична
Виявлення помилок	Під час виконання	Під час виконання	На етапі компіляції
Масштабованість	Середня	Висока	Дуже висока
Інтеграція із React/Next.js	Рідна	Складна	Відмінна (повна підтримка)

Вибір TypeScript для проєкту «EduSteam» зумовлений потребою в суворій типізації даних. Оскільки платформа оперує складними структурами (бали досвіду, рівні, часові інтервали), використання TypeScript дозволяє мінімізувати кількість логічних помилок ще на етапі написання коду, що критично важливо для стабільної роботи алгоритмів гейміфікації [13].

2.3.2 Вибір середовища розробки та допоміжного інструментарію

Для забезпечення високої швидкості написання коду та зручності відлагодження системи було проведено аналіз популярних середовищ розробки (IDE) та текстових редакторів (табл. 2.2) [14].

Таблиця 2.2 – Порівняння інструментарію розробника

Критерій порівняння	Sublime Text	WebStorm	Visual Studio Code
Вартість	Платна (Shareware)	Платна (Ліцензія)	Безкоштовно
Споживання ресурсів	Дуже низьке	Високе	Середнє
Підтримка TypeScript	Через плагіни	Вбудована	Найкраща (рідна)
Екосистема розширень	Середня	Велика	Найбільша у світі

Основним інструментом обрано Visual Studio Code. Його архітектура дозволяє інтегрувати всі необхідні термінали, налагоджувачі та розширення для роботи з Prisma ORM та Git в одному вікні, що значно пришвидшує процес розробки [14].

2.3.3 Обґрунтування вибору системи управління базами даних та ORM

Для зберігання даних освітньої платформи розглянуто реляційні (SQL) та нереляційні (NoSQL) рішення. Порівняльний аналіз наведено у таблиці 2.3.

Таблиця 2.3 – Порівняльний аналіз СУБД

Критерій порівняння	MongoDB (NoSQL)	MySQL (SQL)	PostgreSQL (SQL)
Цілісність даних	Низька	Середня	Дуже висока
Підтримка транзакцій	Обмежена	Повна	Повна (ACID)
Обробка складних реляційних зв'язків	Неефективно	Добре	Відмінно
Робота з JSON	Вбудована	Обмежена	Повна (тип JSONB)

Для реалізації «EduSteam» обрано PostgreSQL. Оскільки освітня платформа базується на чітких зв'язках (користувач — курс — досягнення), реляційна модель є найбільш доцільною. Додатково використано Prisma ORM, що забезпечує автоматичну генерацію TypeScript-типів на основі схеми БД, усуваючи ризик невідповідності даних між сервером та базою [3]-[5].

2.4 Вимоги до технічних і програмних засобів

2.4.1 Необхідні технічні засоби для розробки та експлуатації

Для успішного розгортання та роботи платформи «EduSteam» встановлено такі вимоги до апаратного забезпечення:

- для сервера (хостингу). Мінімум 1 ГБ оперативної пам'яті та 1 ядро процесора (стандартний рівень хмарних сервісів типу Vercel або Docker-контейнерів);
- для розробника. Процесор класу Intel Core i5 або вище, 8 Гб ОЗП, SSD-накопичувач для швидкої компіляції TypeScript;
- для кінцевого користувача. Будь-який сучасний пристрій (ПК, смартфон) з підтримкою сучасних стандартів веббраузерів.

2.4.2 Вимоги до апаратного та програмного забезпечення

Програмні вимоги включають наявність середовища виконання Node.js версії 20.x або вище. Клієнтська частина повинна стабільно працювати в браузерах на базі двигуна Chromium (Google Chrome, Microsoft Edge, Brave), а також у Firefox та Safari останніх версій [15].

2.5 Висновки за розділом 2

У другому розділі було обґрунтовано архітектурні рішення вебплатформи «EduSteam». Обрана трирівнева архітектура із застосуванням Server-Side Rendering дозволяє досягти балансу між швидкістю роботи та складністю бізнес-логіки. Детально розглянуто процеси взаємодії компонентів та алгоритми гейміфікації. Обґрунтовано вибір технологічного стеку: Next.js, TypeScript та PostgreSQL, як найбільш відповідного вимогам безпеки та продуктивності сучасних освітніх систем. Визначено технічні вимоги, що забезпечують стабільну експлуатацію

продукту. Описані архітектурні принципи стають основою для безпосередньої програмної реалізації, яка буде розглянута в наступному розділі [1]-[5].

3 РОЗРОБКА ПРОГРАМИ

3.1 Архітектура системи

Програмна реалізація платформи «EduSteam» базується на сучасній парадигмі компонентно-орієнтованої розробки. Використання Next.js дозволяє об'єднати клієнтську та серверну частини в єдину екосистему, що спрощує типізацію та обмін даними [1].

3.1.1 Структурна схема системи

Структурно система складається з набору взаємопов'язаних модулів, кожен з яких відповідає за певну бізнес-логіку. Основні модулі включають:

- модуль автентифікації: забезпечує реєстрацію, вхід та захист маршрутів;
- модуль контенту: відповідає за рендеринг курсів, уроків та описів;
- ядро гейміфікації: проводить розрахунок очок досвіду (XP) та перевірку досягнень;
- шар доступу до даних (DAL): використовує Prisma Client для виконання запитів до бази PostgreSQL [4]-[5].

Завдяки використанню App Router у Next.js, структура папок проєкту дзеркально відображає архітектуру маршрутів, що робить систему легкою для навігації та масштабування [1].

Структурну схему архітектури вебплатформи відображено на рисунку 3.1.

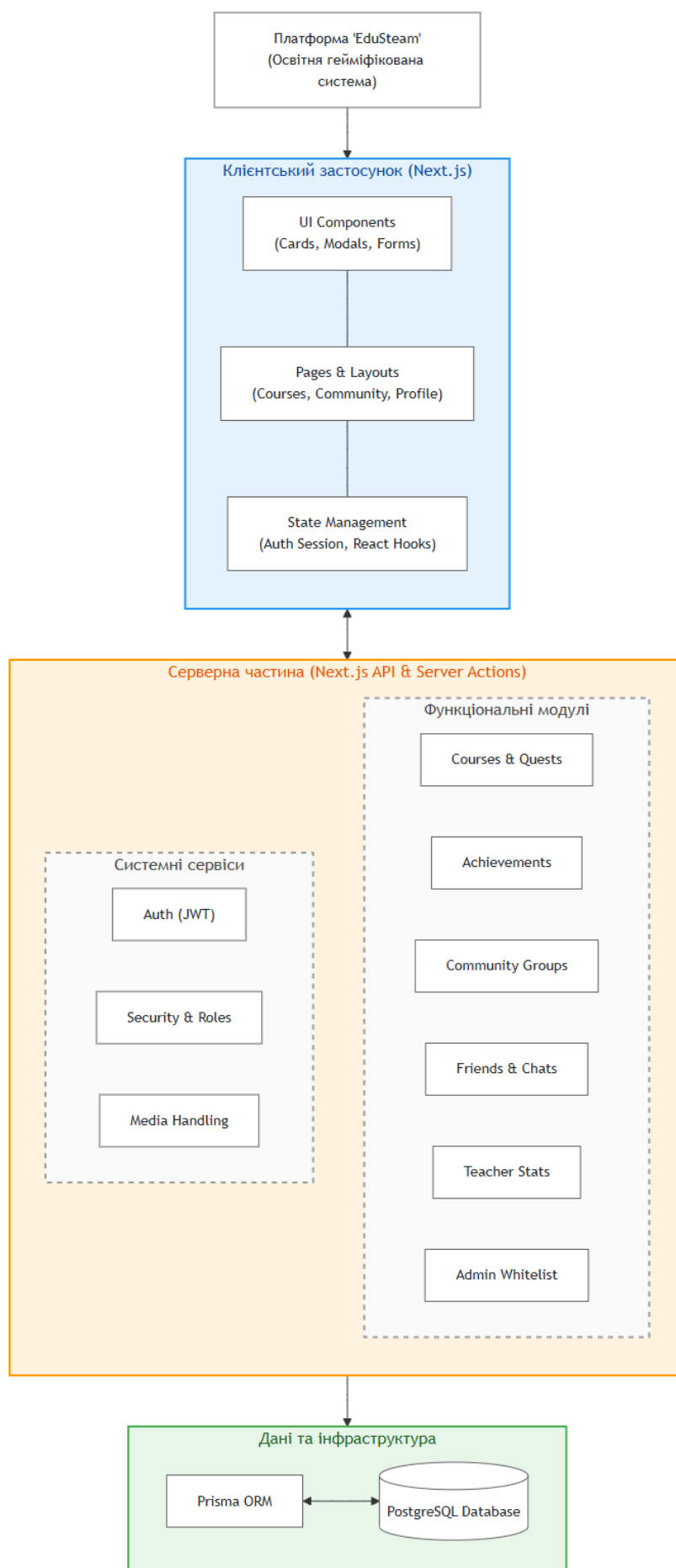


Рисунок 3.1 – Структурна схема архітектури вебплатформи

3.2 Функціонування системи

У цьому підрозділі розглянемо механізми, які забезпечують динамічну роботу платформи та безпеку користувацьких даних.

3.2.1 Функціональна схема (робота з JWT-токенами та маршрутизацією)

Для забезпечення безпеки та підтримки стану авторизації без зберігання сесій на сервері (stateless), у системі реалізовано механізм JSON Web Tokens (JWT) [16].

Процес функціонує за такою схемою:

- запит на вхід: користувач надсилає логін та пароль;
- валідація: сервер перевіряє хеш пароля в базі даних;
- генерація токена: у разі успіху створюється зашифрований токен, що містить userId та role;
- зберігання: токен передається клієнту через заголовок Set-Cookie з прапорцями httpOnly (захист від крадіжки через JS-скрипти) та secure;
- перевірка (Middleware): при кожному переході на захищену сторінку (наприклад, /dashboard), проміжне ПЗ розшифровує токен. Якщо він невалідний – відбувається редирект на сторінку логіна.

3.2.2 Опис процесу обміну даними у системі

Обмін даними між клієнтом і сервером оптимізовано за допомогою Server Actions (нова функція Next.js). Це дозволяє уникнути написання класичних fetch-запитів на фронтенді:

- коли студент натискає кнопку «Завершити квест», викликається серверна функція;

- функція виконує транзакцію в базі даних: додає запис про проходження квесту та нараховує XP користувачу (рис. 3.2);
- після успішного завершення сервер ініціює `revalidatePath`, що змушує клієнтську частину миттєво оновити дані про рівень та прогрес без перезавантаження сторінки.

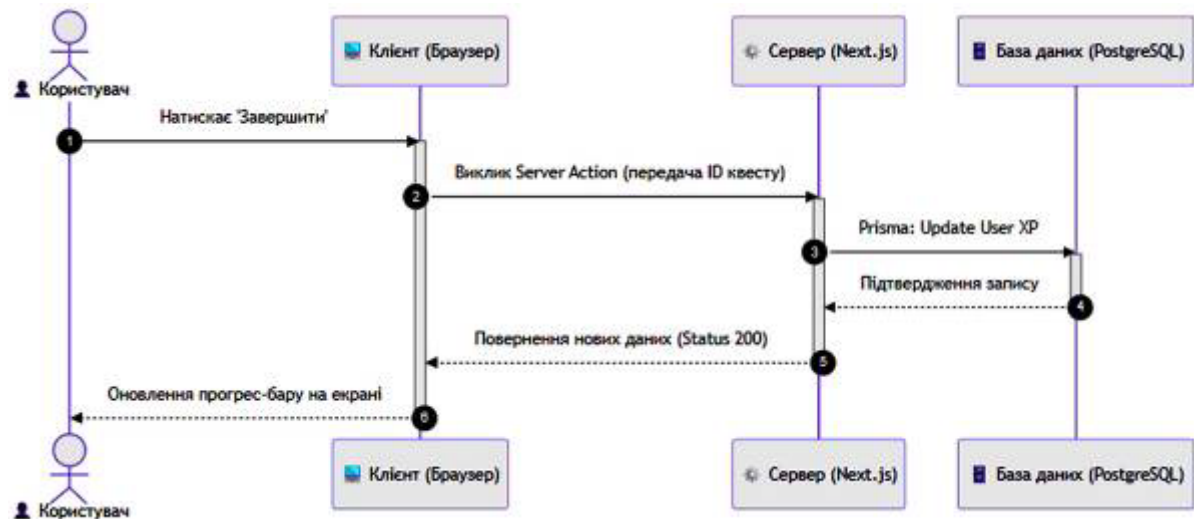


Рисунок 3.2 – Схема процесу обміну даними при нарахуванні ігрового досвіду

3.3 Проєктування бази даних

Проєктування реляційної моделі даних є критичним етапом, оскільки від цілісності структури залежить коректність роботи всієї системи гейміфікації.

3.3.1 Структура бази даних та її декомпозиція на підсистеми

Оскільки система поєднує в собі функціонал управління навчальним контентом та ядро гейміфікації, базу даних було логічно декомпозовано на 4 підсистеми. Структурну схему інтерактивної навчальної платформи «EduSteam» та її основних підсистем представлено на рисунку 3.3.



Рисунок 3.3 – Структурна схема підсистем інтерактивної навчальної платформи «EduSteam»

Зважаючи на високу складність архітектури, для забезпечення зручності підтримки та масштабування системи, базу даних було логічно декомпозовано на 4 ізольовані доменні області (модулі):

Підсистема адміністрування профілів та користувачів є базовою для функціонування платформи (рис. 3.4).

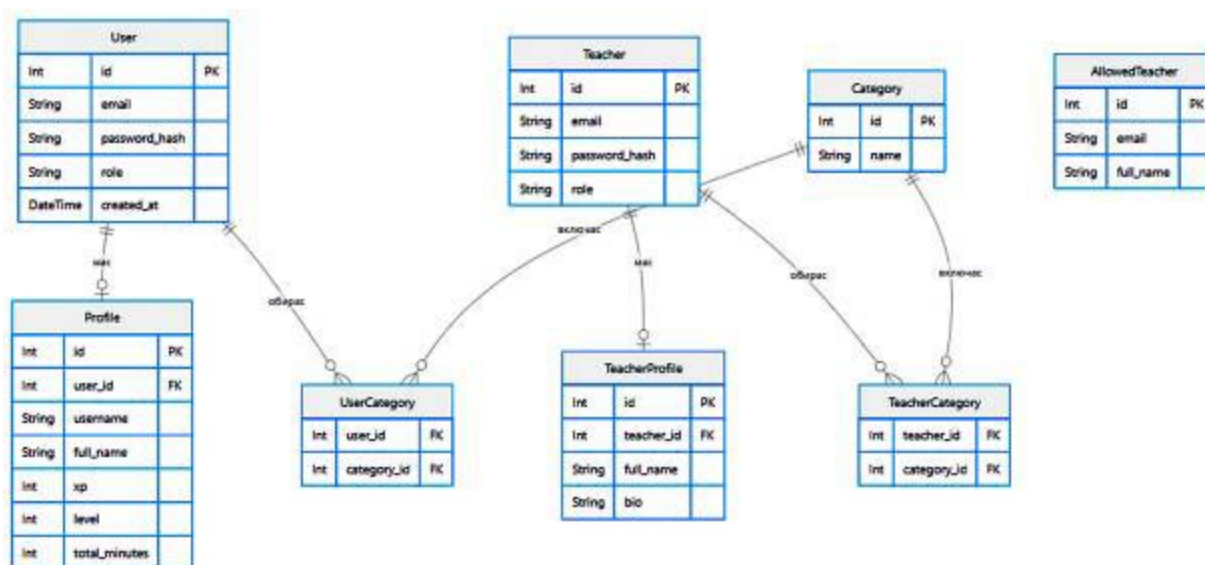


Рисунок 3.4 – ER-діаграма підсистеми адміністрування профілів та користувачів

Тут агрегуються сутності, що ідентифікують користувача в системі, зберігають його облікові дані та базову інформацію профілю.

Ця підсистема дозволяє:

- створювати нових користувачів та зберігати їх облікові дані;
- присвоювати їм відповідні ролі (викладач, студент, адміністратор);
- керувати повноваженнями та привілеями доступу до ресурсів;
- редагувати публічні профілі та налаштування безпеки;
- блокувати користувачів за порушення правил платформи.

Підсистема редагування навчального контенту відповідає за ієрархічне збереження освітніх матеріалів (рис. 3.5).

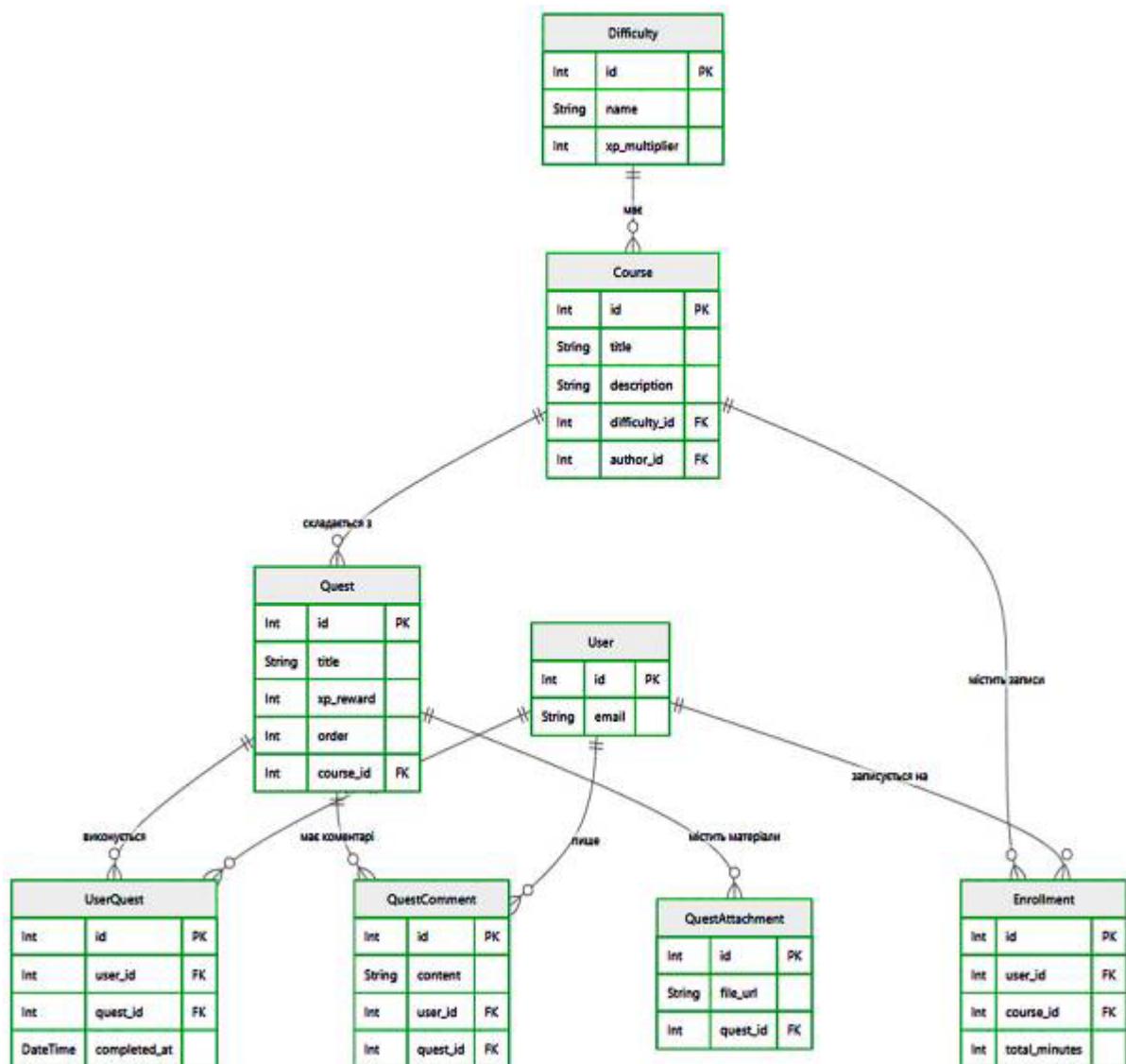


Рисунок 3.5 – ER-діаграма підсистеми редагування навчального контенту

Ця підсистема дозволяє:

- створювати нові навчальні курси та налаштовувати їх опис;
- формувати структуру курсу (модулі, розділи, лекції);
- прикріплювати до уроків мультимедійні файли та текстовий контент;
- зв'язувати створені матеріали з авторами (викладачами);
- фіксувати факт запису студентів на відповідні курси.

До складу модуля входять таблиці, що описують структуру курсів, їхні розділи, прикріплені матеріали та медіадані, а також зв'язки з авторами (викладачами).

Підсистема гейміфікації та нарахування досягнень Реалізує мотиваційні ігрові механіки системи (рис. 3.6).

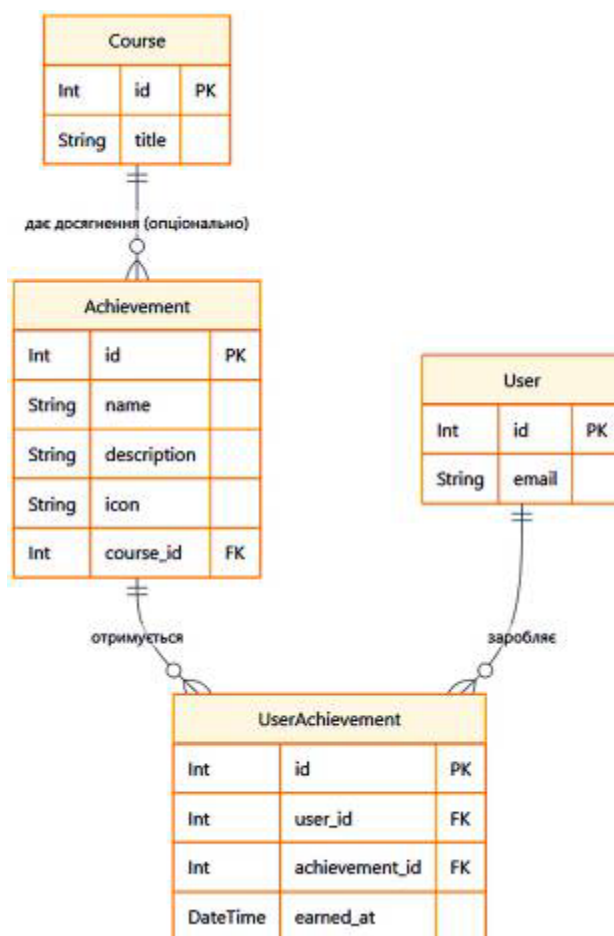


Рисунок 3.6 – ER-діаграма підсистеми гейміфікації та нарахування досягнень

Ця підсистема дозволяє:

- створювати практичні квести та тести для перевірки знань;
- автоматично нараховувати бали досвіду (XP) за успішне виконання завдань;
- формувати каталог віртуальних нагород (досягнень) та умов їх отримання;
- відслідковувати рівень прогресу кожного студента;
- фіксувати історію розблокованих користувачем досягнень.

Таблиці цього модуля відповідають за нарахування балів досвіду, зберігання умов для отримання віртуальних нагород та фіксацію здобутих користувачем досягнень.

Підсистема соціальної взаємодії спільноти забезпечує комунікацію між учасниками освітнього процесу. Містить сутності, які відповідають за обмін повідомленнями, відгуки до курсів, коментарі та систему сповіщень (рис. 3.7).

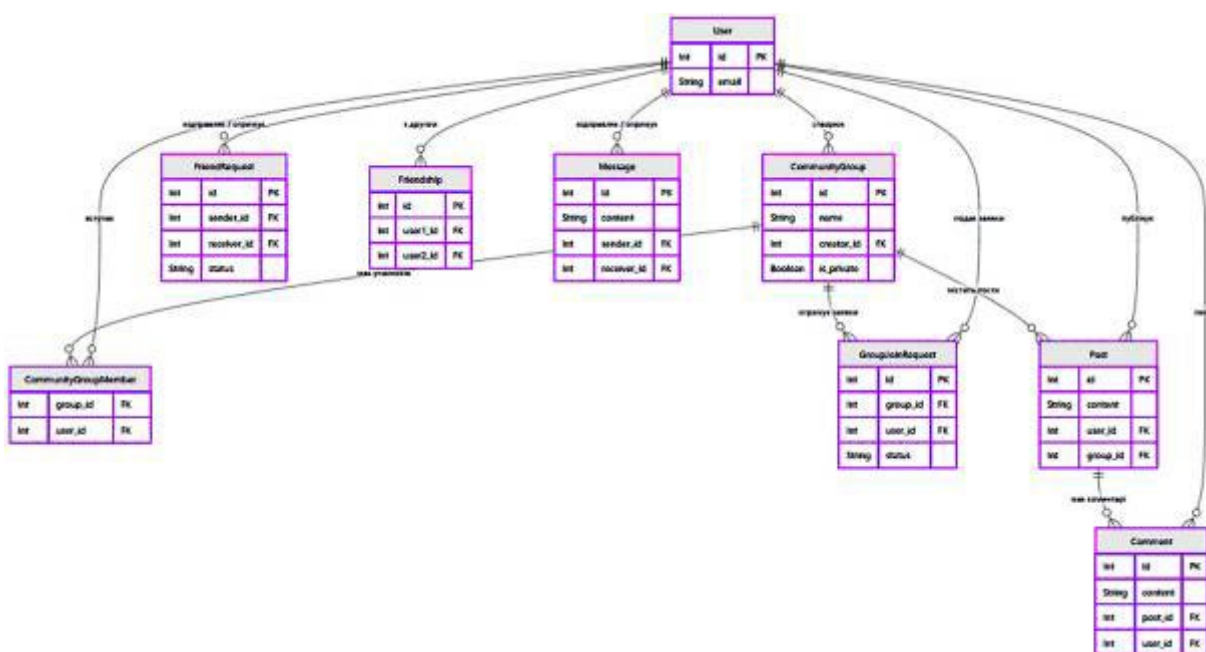


Рисунок 3.7 – ER-діаграма підсистеми соціальної взаємодії спільноти

Ця підсистема дозволяє:

- обмінюватися повідомленнями між користувачами платформи;
- залишати відгуки та оцінки до пройдених навчальних курсів;

- публікувати коментарі до окремих лекцій чи завдань;
- формувати систему автоматичних сповіщень про нові події;
- підтримувати взаємодію викладачів зі студентськими групами.

3.3.2 Взаємозв'язки між таблицями

Зв'язки в базі даних побудовані за принципом реляційної цілісності:

- One-to-One: між User та Profile (у кожного користувача лише один профайл);
- One-to-Many: між Course та Quest (один курс містить багато завдань);
- Many-to-Many: між User та Achievement через проміжну таблицю.

Використання каскадного видалення (onDelete: Cascade) гарантує, що при видаленні користувача всі його прогреси та досягнення також будуть видалені, що запобігає появі «сміттєвих» даних у системі.

3.4 Висновки за розділом 3

У третьому розділі було детально розглянуто програмну реалізацію освітньої платформи. Описано архітектурні модулі системи та логіку їхньої взаємодії. Особливу увагу приділено механізмам безпеки на основі JWT-токенів та сучасним підходам до обміну даними через серверні компоненти Next.js. Також було спроектовано реляційну структуру бази даних, яка забезпечує надійне зберігання інформації про навчальний процес та ігрові досягнення користувачів. Описана програмна логіка є стабільним фундаментом для проведення тестування та експлуатації системи, що буде розглянуто в наступному розділі [16].

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування програми

Розроблена освітня вебплатформа «EduSteam» призначена для розгортання в мережі Інтернет або локальних корпоративних мережах з метою автоматизації та гейміфікації процесу дистанційного навчання. Програмний продукт орієнтований на сучасні заклади освіти, приватні онлайн-школи, а також на незалежних авторів навчальних курсів, які прагнуть підвищити показник завершуваності програм (Retention Rate) серед своїх слухачів [7]-[8].

4.1.1 Огляд планової цільової аудиторії

Ефективність функціонування будь-якої інформаційної системи залежить від чіткого розуміння потреб її кінцевих користувачів. Цільова аудиторія платформи «EduSteam» розділена на три основні сегменти:

а) здобувачі освіти (студенти):

1) вікова категорія: від 14 до 40 років (активні користувачі цифрових технологій);

2) ключові потреби: отримання актуальних знань у зручному форматі, швидкий зворотний зв'язок, динамічна візуалізація власного прогресу, наявність ігрових стимулів для щоденного навчання;

б) творці навчального контенту (викладачі, ментори):

1) ключові потреби: простий та інтуїтивно зрозумілий інструментарій;

2) відсутність потреби у знаннях програмування для налаштування ігрових механік, доступ до статистичних даних щодо активності студентів;

в) адміністратори системи (технічний персонал):

1) ключові потреби: легкість розгортання програмного забезпечення, низькі вимоги до серверних ресурсів, стабільність роботи бази даних при високих навантаженнях;

2) надійний захист персональних даних користувачів [7]-[8].

4.1.2 Візуалізація планованої цільової аудиторії

Для детального аналізу взаємодії користувачів із системою було сформовано матрицю залученості, яка враховує типи мотивації для кожного сегмента аудиторії. Студенти орієнтовані на внутрішню (інтерес до матеріалу) та зовнішню (накопичення ХР, отримання ачивок) мотивацію. Викладачі отримують інструмент для зниження рутинної перевірки завдань завдяки автоматизації квестів.

4.2 Інструкція по роботі з програмою

4.2.1 Встановлення та розгортання програми

Завдяки використанню Full-stack фреймворку Next.js та ORM Prisma, процес розгортання платформи є стандартизованим та не потребує складного адміністрування [1]-[4].

Послідовність дій для локального розгортання проєкту:

- встановити на цільовий комп'ютер середовище виконання Node.js (версії не нижче 20.x) та СУБД PostgreSQL;
- завантажити початковий код програми з репозиторію та перейти в кореневу директорію проєкту через термінал;
- виконати команду інсталяції залежностей: `npm install` (або `bun install`);

– створити в корені проекту файл налаштувань оточення `.env` та вказати в ньому параметри підключення до бази даних та секретний ключ для JWT:

```
DATABASE_URL="postgresql://johndoe:mypassword@localhost:5432/edusteam?schema=public"
```

```
JWT_SECRET="super_secret_key_for_token_encryption"
```

– виконати міграцію структури бази даних за допомогою Prisma:

```
npx prisma migrate dev
```

– за необхідністю запустити скрипт початкового наповнення бази даних (seed):

```
npx prisma db seed
```

4.2.2 Запуск та початкове налаштування системи

Після успішного виконання всіх етапів інсталяції запуск платформи в режимі розробника здійснюється командою:

```
npm run dev
```

За замовчуванням локальний сервер починає роботу за адресою:

```
http://localhost:3000
```

При першому переході за цією адресою новий користувач повинен зареєструвати обліковий запис. Процес реєстрації має чітке розмежування та передбачає вибір відповідної ролі:

– «Студент» для проходження курсів (рис. 4.1);

← На головну

EduSteam

Створити акаунт

Хто ти?


Студент
Проходжу курси та виконую квести


Викладач
Створюю курси та навчаю студентів

Ім'я

Микита|

Прізвище

Пастушенко

Email

nikitkapastusenko319@gmail.com

Пароль

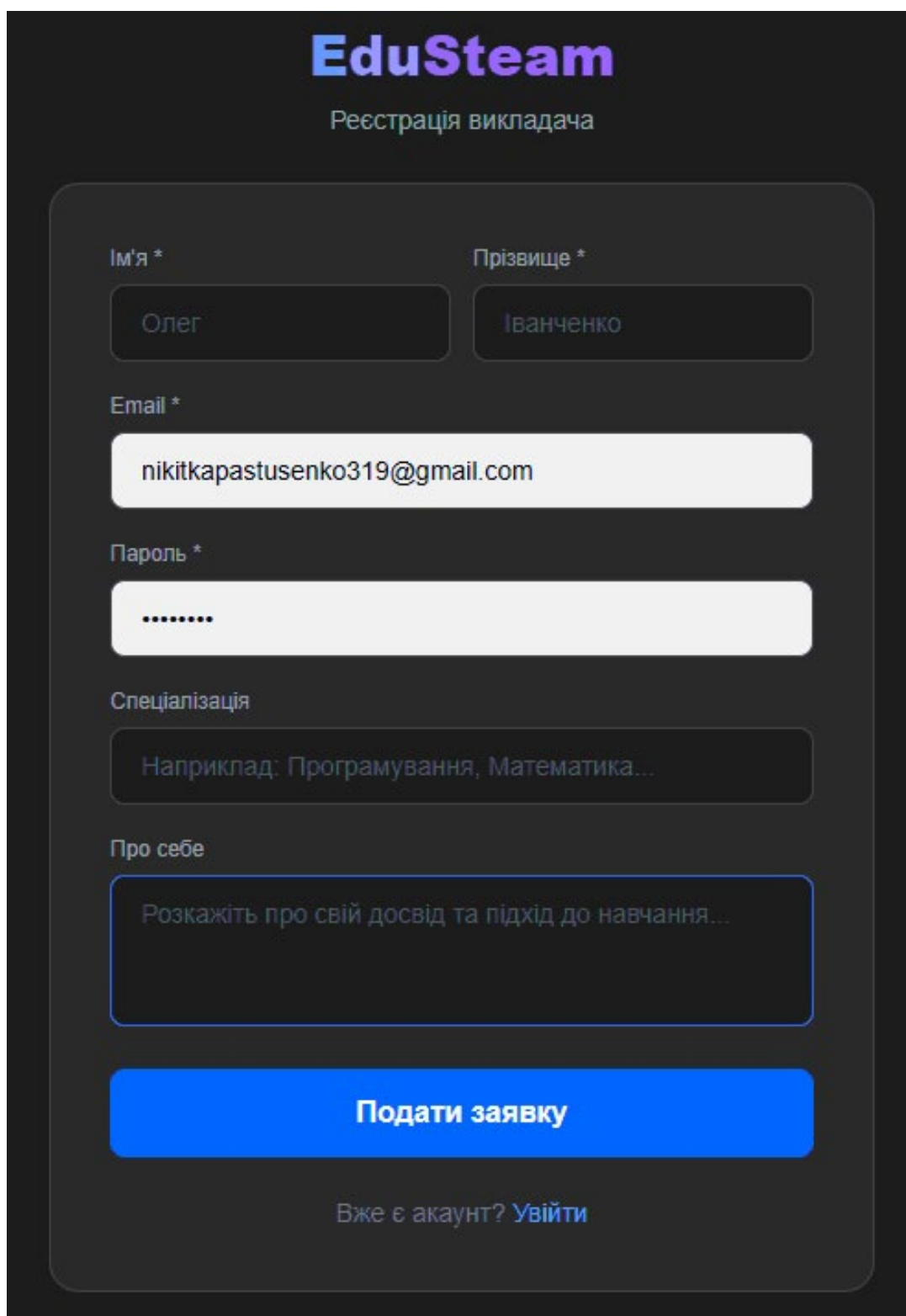
.....

Зареєструватися

Вже є профіль? [Увійти](#)

Рисунок 4.1 – Форма реєстрації облікового запису студента

– «Викладач» для створення навчального контенту (рис. 4.2).



The image shows a registration form for 'EduSteam' titled 'Реєстрація викладача' (Teacher Registration). The form is set against a dark background. It includes fields for 'Ім'я *' (Name) with the value 'Олег', 'Прізвище *' (Surname) with the value 'Іванченко', 'Email *' with the value 'nikitkapastusenko319@gmail.com', and 'Пароль *' (Password) shown as dots. There is a 'Спеціалізація' (Specialization) field with the placeholder 'Наприклад: Програмування, Математика...'. A 'Про себе' (About me) section contains a text area with the placeholder 'Розкажіть про свій досвід та підхід до навчання...'. A large blue button labeled 'Подати заявку' (Submit application) is at the bottom, followed by a link 'Вже є акаунт? Увійти' (Already have an account? Log in).

EduSteam
Реєстрація викладача

Ім'я *
Олег

Прізвище *
Іванченко

Email *
nikitkapastusenko319@gmail.com

Пароль *
.....

Спеціалізація
Наприклад: Програмування, Математика...

Про себе
Розкажіть про свій досвід та підхід до навчання...

Подати заявку

Вже є акаунт? [Увійти](#)

Рисунок 4.2 – Форма реєстрації облікового запису викладача

Після успішної реєстрації та авторизації користувач потрапляє на головну сторінку застосунку, яка виконує роль навігаційного центру та візитної картки освітньої платформи (рис. 4.3).

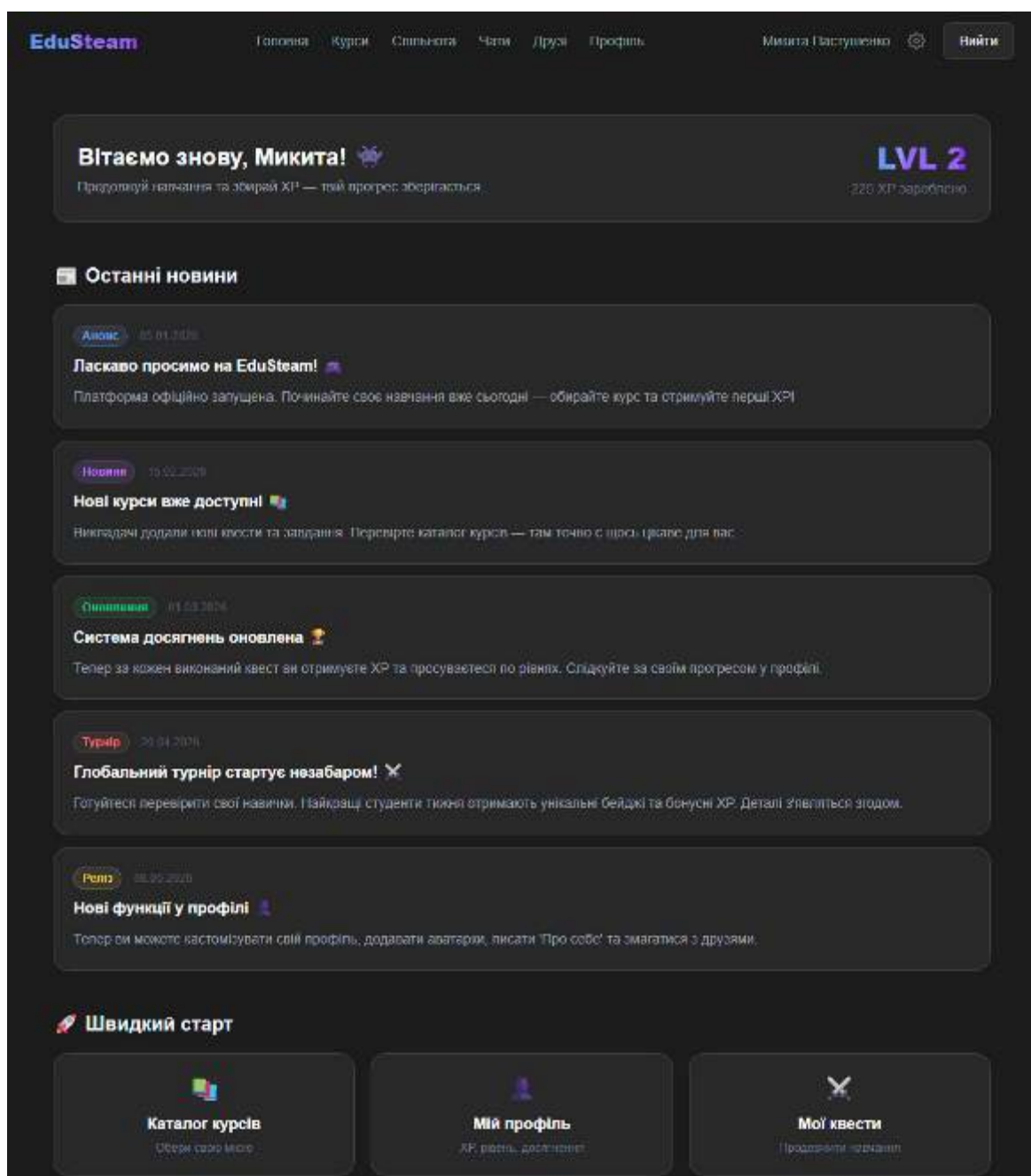


Рисунок 4.3 – Головна сторінка освітньої платформи «EduSteam»

4.2.3 Методика роботи з платформою

Алгоритм дій для користувача з роллю «Викладач»:

- авторизуватися в системі та перейти до розділу «Панель керування курсами»;
- натиснути кнопку «Створити курс», заповнити текстові поля (назва, опис, категорія) та зберегти зміни (рис. 4.4);

Рисунок 4.4 – Форма створення нового навчального курсу викладачем

- усередині створеного курсу додати необхідну кількість навчальних модулів та лекційних занять;
- для кожного модуля створити «Квест» (практичне завдання) та вказати кількість очок досвіду (XP), які студент отримає автоматично після його виконання (рис. 4.5);

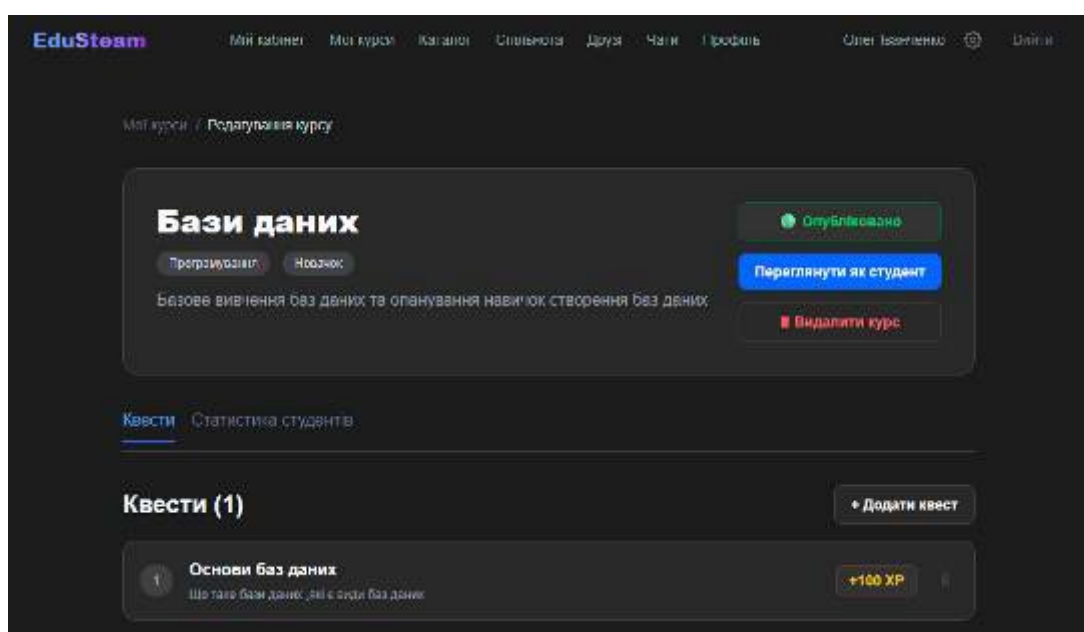


Рисунок 4.5 – Інтерфейс управління курсом та додавання квестів

– відстежувати детальну статистику за кожним курсом: кількість студентів, пройдено квестів, витрачений час та загальний прогрес (рис. 4.6).

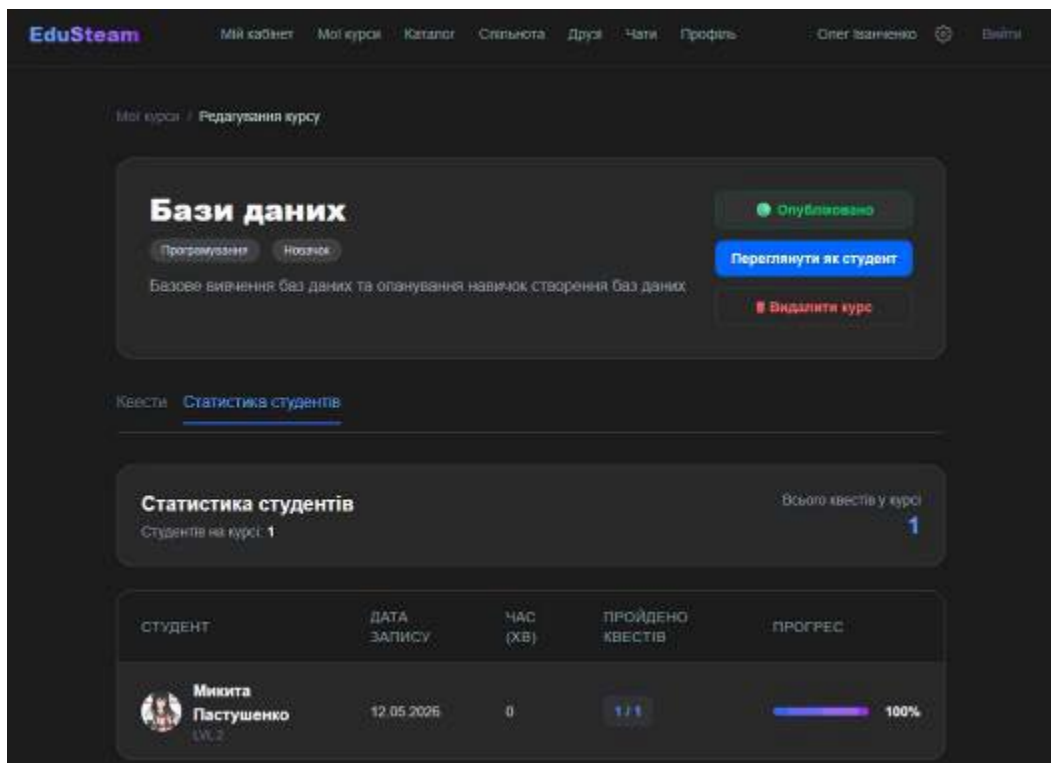


Рисунок 4.6 – Перегляд статистики успішності студентів на курсі

Алгоритм дій для користувача з роллю «Студент»:

- створити профіль або увійти під своїми обліковими даними;
- перейти до глобального «Каталогу курсів» та обрати необхідну навчальну програму (рис. 4.7);
- натиснути кнопку «Записатися на курс», після чого він з’явиться на персональному дашборді;
- почати вивчення лекційного матеріалу. Система в автоматичному режимі веде трекінг часу перебування на сторінці уроку;
- виконати квест та надіслати його на перевірку (або підтвердити виконання). Після тригера успішного завершення баланс ХР у профілі миттєво оновлюється, а при досягненні ліміту – підвищується рівень користувача.

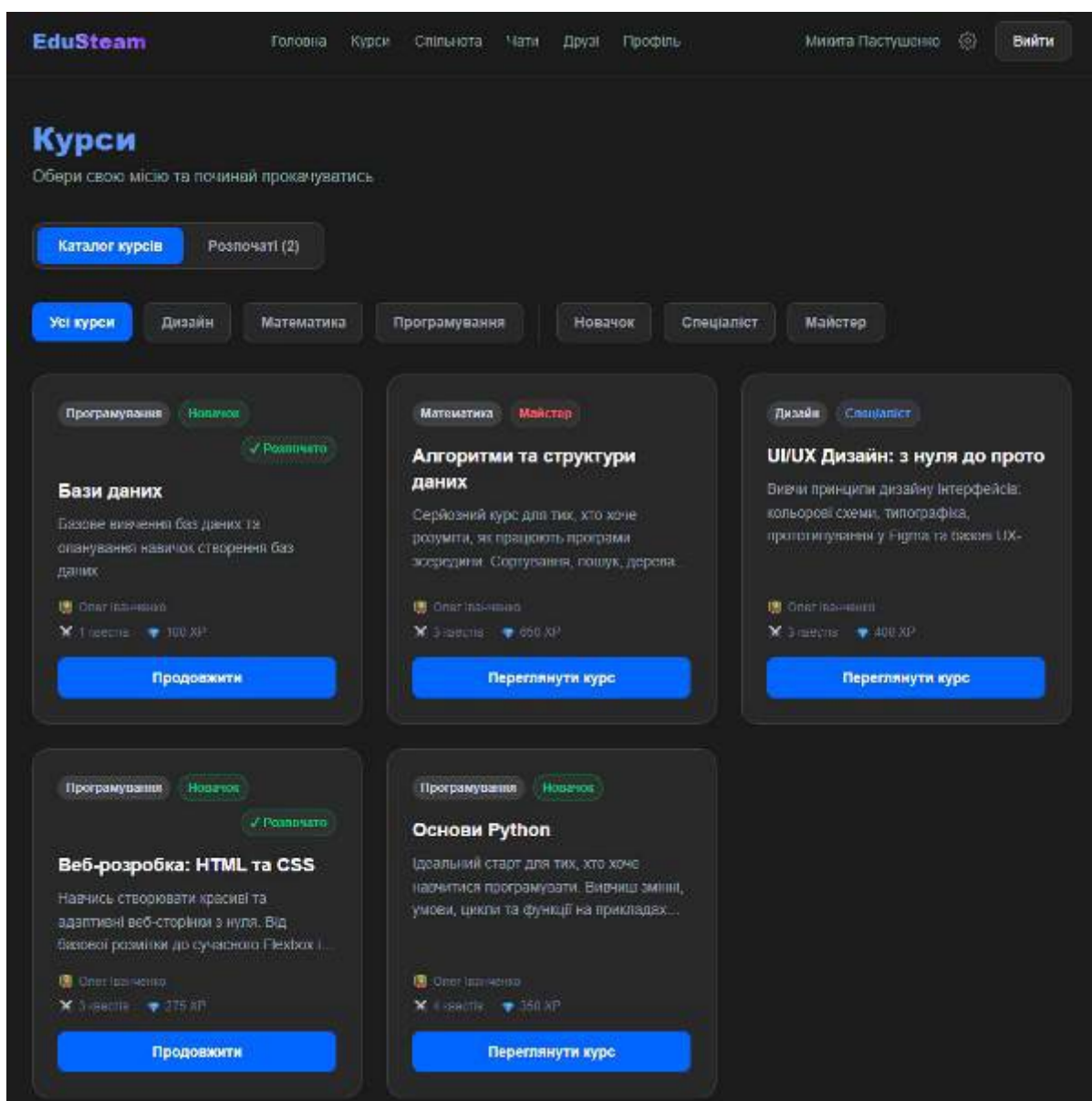


Рисунок 4.7 – Каталог доступних курсів з позиції студента

4.3 Приклад роботи користувача з програмою

Типовий сценарій взаємодії виглядає наступним чином. Студент авторизується в системі через захищену форму. Після входу сервер зчитує його JWT-токен, робить запит до PostgreSQL через Prisma і повертає персоналізовану сторінку. Студент бачить свій поточний 2-й рівень і прогрес-бар, наприклад, «225/400 XP до наступного рівня» (рис. 4.8) [16].

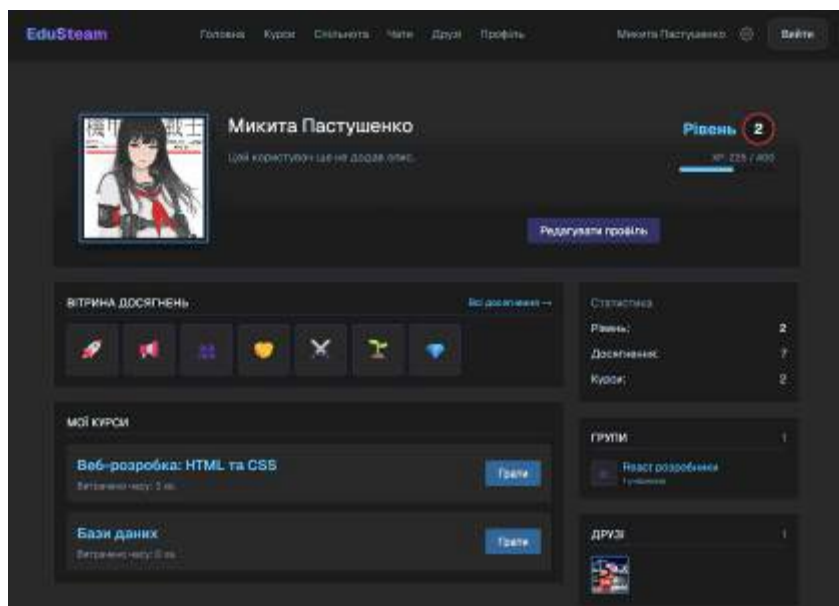


Рисунок 4.8 – Персональний дашборд студента з відображенням ігрового прогресу

Студент відкриває курс «Основи вебпрограмування» та отримує доступ до списку модулів і загальної статистики свого прогресу по цьому курсу (рис. 4.9).

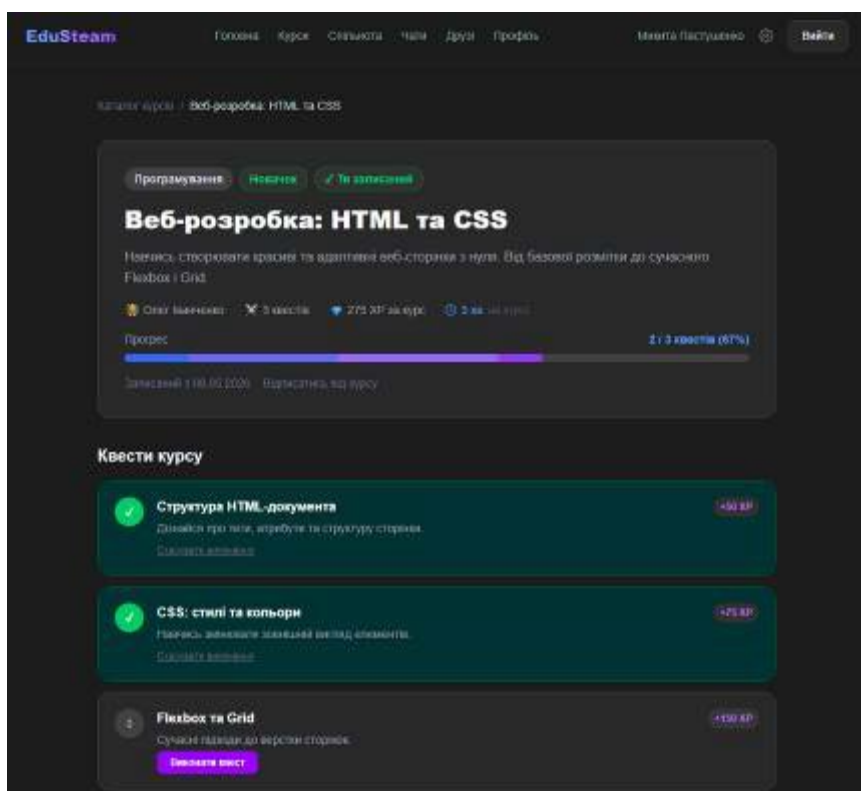


Рисунок 4.9 – Інтерфейс проходження навчального квесту

Студент виконує практичне завдання «Створення першої HTML-сторінки». Після натискання кнопки «Перевірити завдання», серверна функція (Server Action) фіксує успіх, додає 150 XP до профілю. Прогрес-бар на екрані динамічно змінюється на значення 375/400 XP без повного перезавантаження сторінки користувача (рис. 4.10).

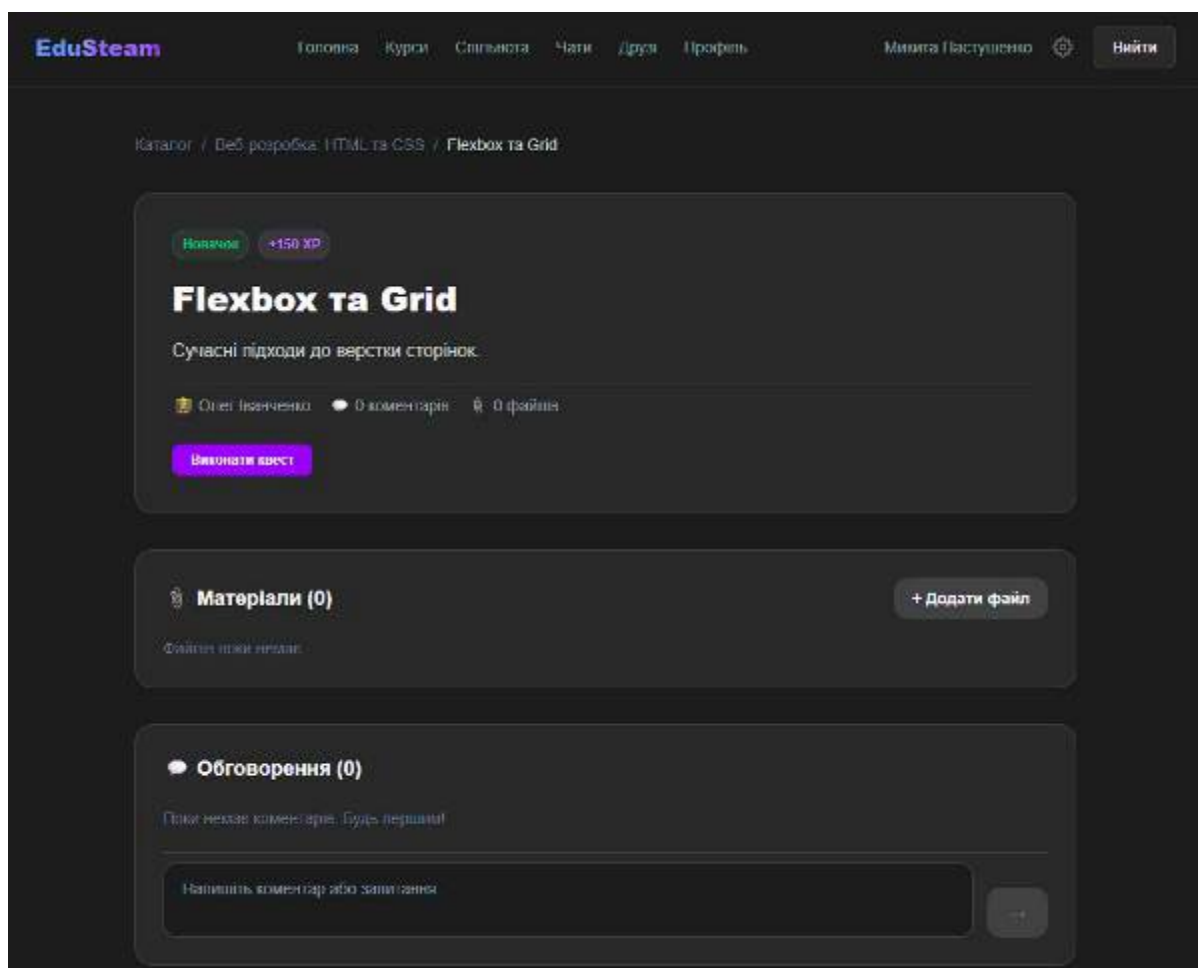


Рисунок 4.10 – Інтерфейс проходження навчального квесту

Окрім накопичення очок досвіду, усі здобуті користувачем віртуальні нагороди та глобальні досягнення зберігаються та візуалізуються у спеціалізованому розділі профілю. Це формує ефект колекціонування та слугує додатковим стимулом до проходження складних курсів (рис. 4.11).

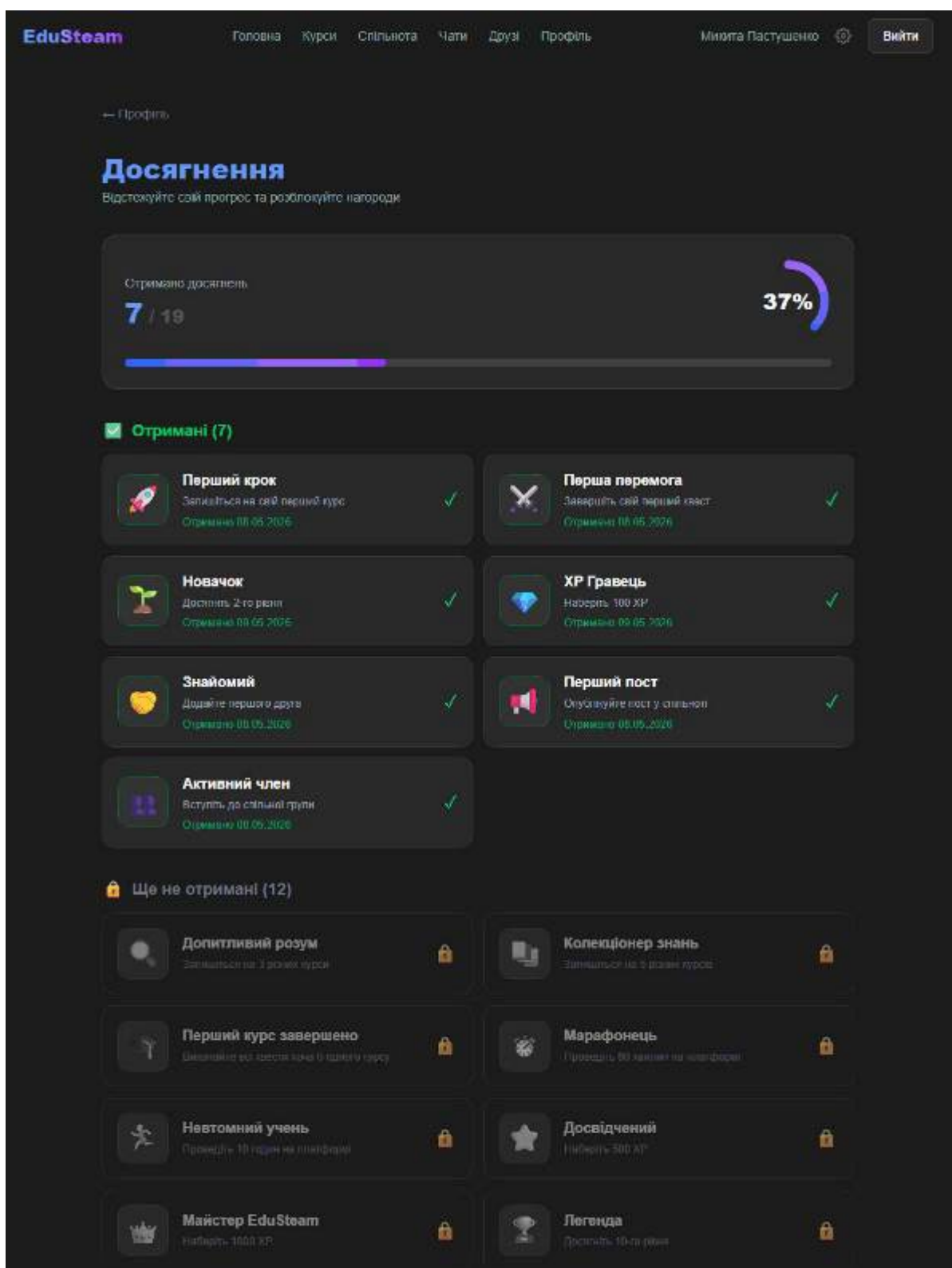


Рисунок 4.11 – Розділ віртуальних досягнень та нагород користувача

4.4 Тестування роботи програми

4.4.1 Функціональне тестування

Функціональне тестування проводилося з метою перевірки відповідності реалізованого програмного забезпечення всім вимогам, сформованим у Розділі 1. Об'єктами тестування виступали модулі автентифікації, логіка розмежування прав доступу, інтерфейс створення курсів та ядро гейміфікації.

Тестування здійснювалося методом «чорної скриньки» (Black Box Testing) шляхом ручного та автоматизованого введення валідних і невалідних даних у форми застосунку.

4.4.2 Аналіз функціонального тестування

Під час проведення функціонального тестування було виявлено та успішно усунуто кілька архітектурних проблем. Зокрема, виникав конфлікт паралельних запитів у Prisma ORM, коли студент одночасно відкривав кілька уроків в окремих вкладках браузера, що призводило до спроби одночасного перезапису поля `totalMinutes` [4].

Проблему було вирішено шляхом впровадження оптимістичного блокування (Optimistic Locking) на рівні бази даних та оптимізації серверних функцій оновлення прогресу. Усі повторні тести підтвердили 100% коректність виконання операцій.

4.4.3 Тестування продуктивності

Для оцінки швидкодії платформи було використано інструменти автоматизованого аналізу вебдодатків (Lighthouse) та проведено стрес-тестування за допомогою імітації одночасної роботи кількох десятків користувачів. Основна увага приділялася швидкості рендерингу сторінок завдяки використанню технології SSR.

4.4.4 Аналіз продуктивності

Аналіз показав, що завдяки серверному рендерингу (Server-Side Rendering) у Next.js, час до першого відтворення контенту (First Contentful Paint, FCP) становить менше 0.4 секунди, що є відмінним показником для високонавантажених освітніх платформ. Час інтерактивності (Time to Interactive, TTI) не перевищує 0.8 секунди (рис. 4.12).

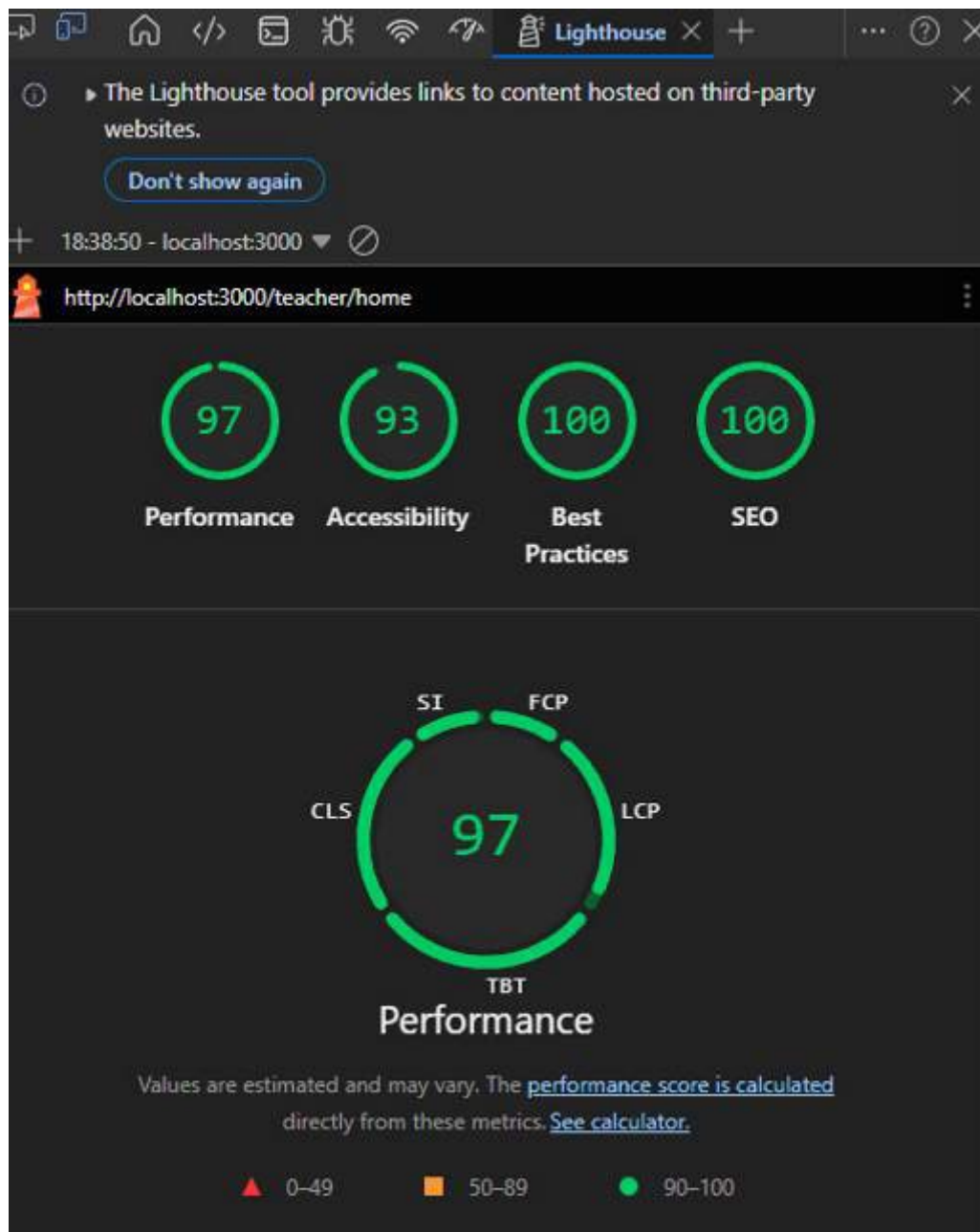


Рисунок 4.12 – Результати автоматизованого тестування продуктивності платформи у Google Lighthouse

Використання індексів у PostgreSQL для полів `userId` та `courseId` дозволило зберегти стабільну швидкість виконання складних вибірок даних навіть при лінійному зростанні кількості записів [5].

4.4.5 Тестування безпеки

Безпека користувацьких даних та захист системи від несанкціонованого доступу перевірялися за кількома напрямками:

- стійкість системи автентифікації до перехоплення сесій;
- неможливість підміни ролі користувача на клієнтській стороні;
- надійність шифрування конфіденційних даних (паролів).

4.4.6 Аналіз безпеки

Тестування безпеки підтвердило надійність обраної архітектури:

- захист паролів: використання бібліотеки `bcrypt` із фактором безпеки (`salt rounds`) рівним 10 унеможливорює розшифрування паролів навіть у разі повного витоку зліпку бази даних;
- захист сесій: оскільки JWT-токени зберігаються в `HTTP-only cookies`, спроби викрасти сесію через зловмисні JavaScript-скрипти (XSS-атаки) виявилися безуспішними – браузер блокує доступ до цих кукі-файлів для будь-яких скриптів [16];
- контроль доступу: спроби отримати доступ до панелі викладача за прямим посиланням (наприклад, `/admin/create-course`) з облікового запису студента повністю блокуються на рівні `Next.js Middleware`, яке повертає сторінку помилки 403 (Forbidden).

4.4.7 Аналіз користувацького досвіду (UI/UX)

На етапі фінального тестування було проведено експертну оцінку інтерфейсу. Завдяки використанню компонентної бібліотеки та адаптивної сітки Tailwind CSS, інтерфейс платформи демонструє високу гнучкість. Платформа є однаково зручною як для десктопних моніторів (де викладачу важливо бачити великі таблиці керування контентом), так і для екранів мобільних телефонів (де студент проходить лекції та переглядає свої досягнення в дорозі) [17].

Окрему увагу під час проєктування інтерфейсу було приділено підсистемі соціальної взаємодії. Для того щоб студенти не відчували ізольованості під час дистанційного навчання, інтерфейси сторінок «Друзі» та «Спільнота» спроектовано за патернами сучасних соціальних мереж (рис. 4.13 та 4.14).

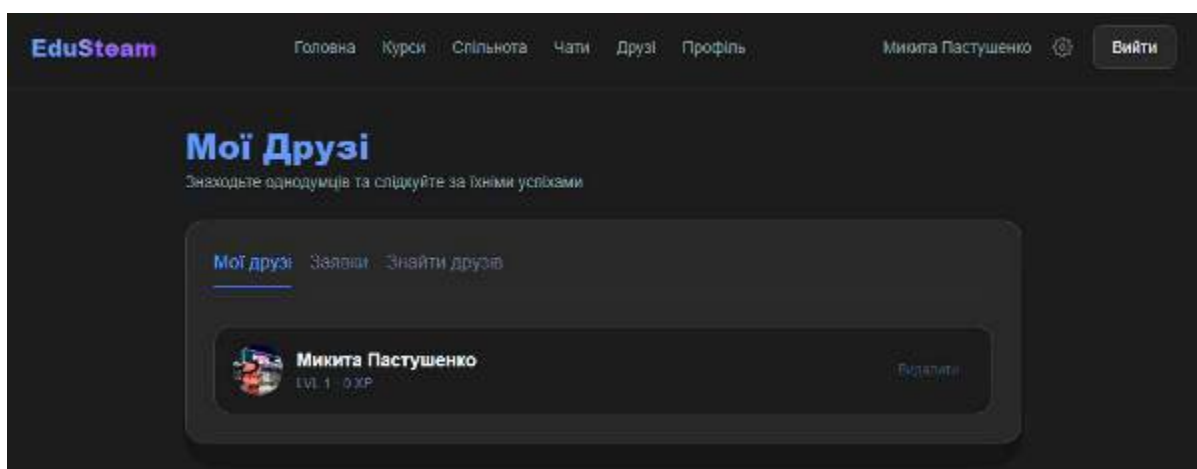


Рисунок 4.13 – Інтерфейс розділу соціальної взаємодії (Друзі)

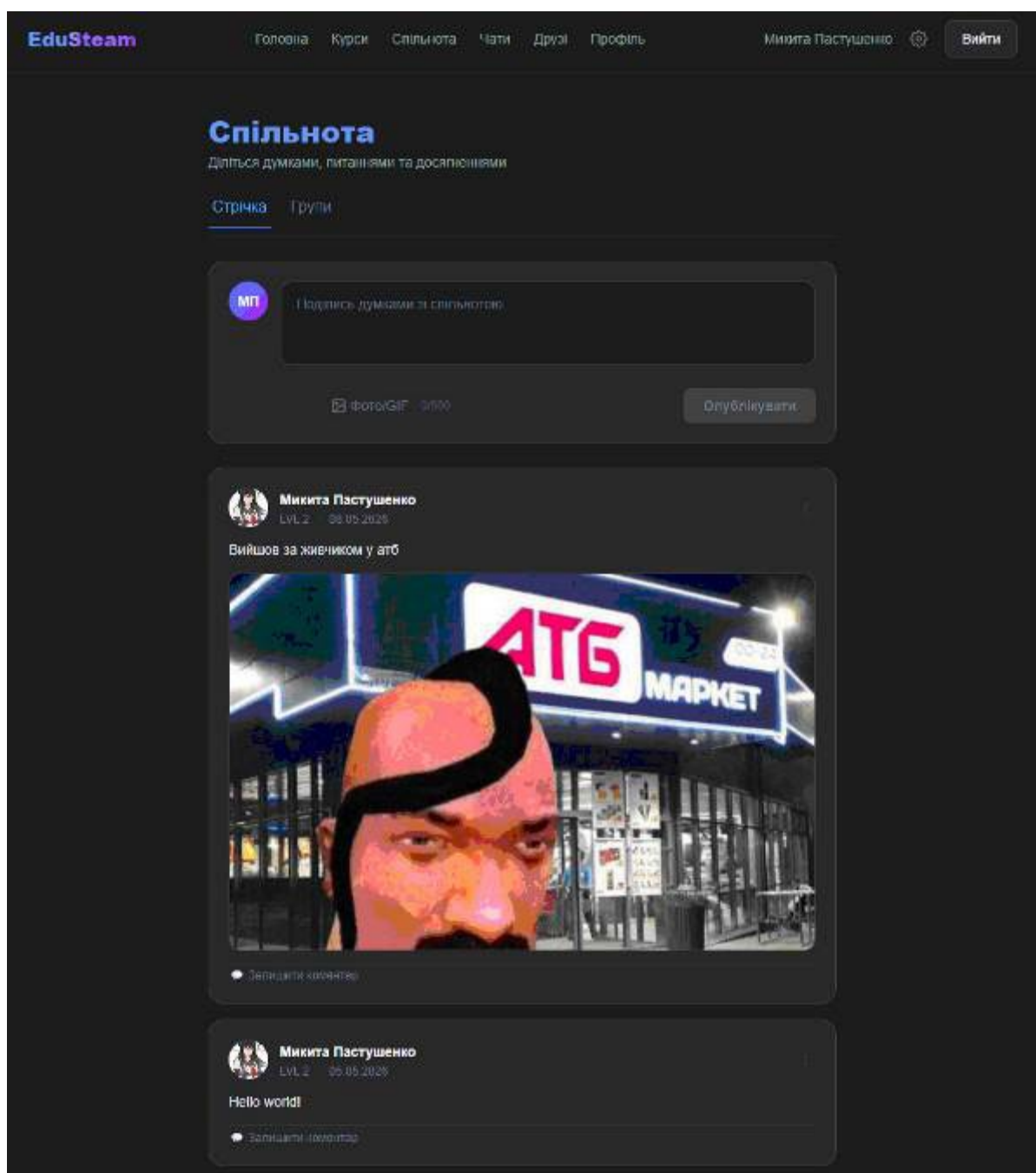


Рисунок 4.14 – Інтерфейс розділу соціальної взаємодії (Спільнота)

Крім того, реалізовано ергономічний інтерфейс системи приватних повідомлень (чатів), що дозволяє студентам та викладачам миттєво обмінюватися інформацією, не покидаючи навчальне середовище (рис. 4.15).

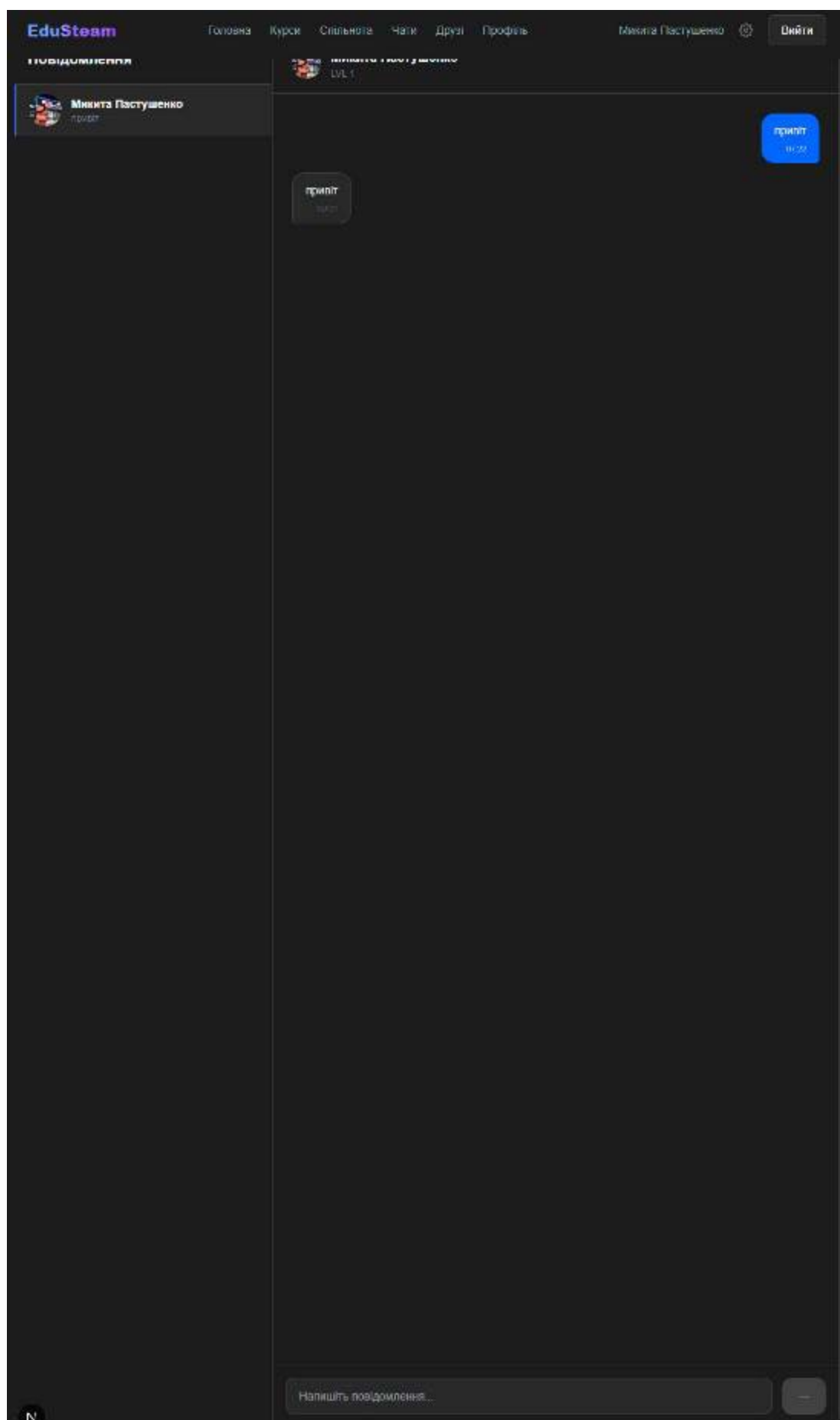


Рисунок 4.15 – Інтерфейс системи обміну приватними повідомленнями

4.5 Рекомендації щодо використання

Для стабільної та довгострокової експлуатації вебплатформи «EduSteam» рекомендується дотримуватися наступних технічних регламентів:

- хостинг: для розгортання клієнт-серверної частини ідеально підходять хмарні сервіси з підтримкою Next.js (наприклад, Vercel або власні VPS-сервери під керуванням Docker) [12];
- база даних: як хмарне рішення для PostgreSQL рекомендується використовувати сервіси з автоматичним масштабуванням (наприклад, Neon або Supabase), що дозволить системі не падати під час масових екзаменаційних сесій чи синхронних запусків курсів;
- резервне копіювання: необхідно налаштувати щоденне автоматичне створення бекапів (pg_dump) таблиць бази даних, особливо таблиць Profile та UserAchievement, оскільки втрата ігрового прогресу користувачів веде до повної втрати їхньої мотивації [18]-[19].

4.6 Плани з розвитку

Розроблена система має модульну архітектуру, що відкриває широкі можливості для її подальшої модернізації. У якості пріоритетних напрямків розвитку проєкту «EduSteam» визначено:

- лідерборди (Турнірні таблиці): додавання щотижневих таблиць лідерів серед студентів однієї спеціальності для стимулювання здорової конкуренції (змагальний елемент гейміфікації);
- серії безперервності (Streaks): впровадження механіки відстеження регулярності відвідувань (як у Duolingo), коли за щоденне вивчення хоча б однієї лекції студент отримує додаткові множники досвіду (XP multipliers);

– інтеграція ШІ: додавання модуля штучного інтелекту для автоматичного генерування унікальних квестів (тестів та практичних завдань) на основі завантаженого викладачем текстового контенту лекції.

ВИСНОВКИ

У рамках виконання дипломної роботи було успішно вирішено актуальне науково-практичне завдання – проєктування та створення інтерактивної освітньої вебплатформи «EduSteam». Здобуті результати переконливо доводять дієвість обраного підходу: розроблений програмний продукт ефективно поєднує класичні інструменти дистанційного навчання із сучасними механіками гейміфікації, що дозволяє суттєво підвищити рівень мотивації та утримання уваги здобувачів освіти.

Аналіз предметної області та огляд конкурентного середовища підтвердили наявність концептуальної прогалини на ринку EdTech. Встановлено, що традиційні академічні системи управління навчанням (наприклад, Moodle) характеризуються надмірною консервативністю та відсутністю вбудованих мотиваційних тригерів, тоді як суто ігрові застосунки (Duolingo) не дозволяють незалежним викладачам створювати власні комплексні курси. Це стало вагомим обґрунтуванням для розробки гібридного рішення, яке б пропонувало викладачам гнучкість управління контентом, а студентам – ігровий досвід у процесі навчання.

Інженерним фундаментом системи стала сучасна компонентна архітектура на базі фреймворку Next.js, який забезпечив обробку як клієнтської, так і серверної логіки. Використання технології Server-Side Rendering (SSR) дозволило досягти миттєвого відображення навчальних матеріалів. Інтеграція об'єктно-реляційного відображення Prisma ORM у зв'язці з надійною СУБД PostgreSQL забезпечила створення суворо типізованої та цілісної структури даних, що є критично важливим для безпомилкового збереження інформації про складні зв'язки між ігровими досягненнями, курсами та прогресом профілів.

На етапі програмування створено повноцінний набір функціональних модулів: середовище створення лекцій, систему реєстрації на курси, механізми трекінгу витраченого часу та модуль генерації квестів.

Унікальною перевагою «EduSteam» є алгоритмічне ядро гейміфікації, яке за допомогою Next.js Server Actions автоматично розраховує очки досвіду (XP), ініціює переходи на нові рівні та присвоює віртуальні нагороди без жодних перезавантажень сторінки користувача. Також реалізовано чітку рольову модель, що відділяє адміністративні можливості викладача від робочого простору студента.

Експериментальне дослідження та комплексне тестування підтвердили високу стабільність платформи. Аналіз продуктивності (за допомогою інструментів Lighthouse) зафіксував швидкість відтворення контенту (FCP) на рівні менше ніж 0,4 секунди. Виявлені під час розробки архітектурні ускладнення, зокрема конфлікти паралельного запису даних у Prisma під час одночасного виконання кількох квестів, були успішно вирішені шляхом оптимізації транзакцій, що забезпечило готовність системи до високих навантажень.

Питання конфіденційності та захисту інформації вирішено шляхом впровадження сучасних криптографічних стандартів. Процес автентифікації спирається на використання JSON Web Tokens (JWT), які безпечно зберігаються у HTTP-only cookies, унеможливорюючи XSS-атаки. Крім того, паролі користувачів надійно хешуються алгоритмом bcrypt, а проміжне програмне забезпечення (Middleware) суворо контролює маршрутизацію, блокуючи несанкціонований доступ до панелей адміністрування.

Оцінка ринкового потенціалу свідчить, що розроблена система становить високу практичну цінність для приватних репетиторів, спеціалізованих ІТ-шкіл та корпоративних центрів підвищення кваліфікації. Можливість легкого розгортання та відсутність потреби у платному ліцензуванні робить «EduSteam» привабливим рішенням для малого та середнього освітнього бізнесу.

Визначено стратегічні вектори подальшої еволюції платформи, серед яких інтеграція алгоритмів штучного інтелекту для автоматичної

генерації тестових завдань, створення змагальних таблиць лідерів (Leaderboards) та впровадження механік щоденних серій навчання (Streaks). Створений вебзастосунок є повністю працездатним продуктом, який вирішує проблему низької залученості студентів та формує надійну базу для розвитку повномасштабної цифрової освітньої екосистеми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Next.js Documentation. Vercel. URL: <https://nextjs.org/docs> (date of access: 15.05.2026).
2. React Documentation. Meta Platforms, Inc. URL: <https://react.dev> (date of access: 15.05.2026).
3. TypeScript. Microsoft Corporation. URL: <https://www.typescriptlang.org> (date of access: 15.05.2026).
4. Prisma ORM Documentation. Prisma Data, Inc. URL: <https://www.prisma.io/docs> (date of access: 15.05.2026).
5. PostgreSQL Documentation. The PostgreSQL Global Development Group. URL: <https://www.postgresql.org/docs> (date of access: 15.05.2026).
6. Kapp K. M. The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education. San Francisco : Pfeiffer, 2012. 336 p.
7. Ryan R. M., Deci E. L. Self-Determination Theory: Basic Psychological Needs in Motivation, Development, and Wellness. New York : Guilford Press, 2017. 756 p.
8. Bartle R. Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD research*. 1996. Vol. 1, No. 1. P. 19.
9. Coursera. Coursera Inc. URL: <https://www.coursera.org> (date of access: 15.05.2026).
10. Duolingo. Duolingo Inc. URL: <https://www.duolingo.com> (date of access: 15.05.2026).
11. Moodle – Open-source learning platform. Moodle Pty Ltd. URL: <https://moodle.org> (date of access: 15.05.2026).
12. Server Actions and Mutations. Vercel. URL: <https://nextjs.org/docs/app/building-your-application/data-fetching/server-actions-and-mutations> (date of access: 15.05.2026).

13. TypeScript Handbook. Microsoft Corporation. URL: <https://www.typescriptlang.org/docs/handbook> (date of access: 15.05.2026).
14. Visual Studio Code. Microsoft Corporation. URL: <https://code.visualstudio.com> (date of access: 15.05.2026).
15. Chrome DevTools: Lighthouse. Google Developers. URL: <https://developer.chrome.com/docs/lighthouse/overview> (date of access: 15.05.2026).
16. JSON Web Tokens. Auth0 Inc. URL: <https://jwt.io/introduction> (date of access: 15.05.2026).
17. Tailwind CSS Documentation. Tailwind Labs, Inc. URL: <https://tailwindcss.com/docs> (date of access: 15.05.2026).
18. bcrypt. npm, Inc. URL: <https://www.npmjs.com/package/bcrypt> (date of access: 15.05.2026).
19. Content Security Policy (CSP). Mozilla Developer Network. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP> (date of access: 15.05.2026).

ДОДАТОК А
Технічне завдання

Вступ

Освітня вебплатформа «EduSteam» створюється з метою модернізації процесів дистанційного навчання шляхом глибокої інтеграції ігрових механік (гейміфікації). Програмний продукт орієнтований на заклади вищої освіти, приватні онлайн-школи та незалежних менторів, які стикаються з проблемою низької мотивації слухачів. Цільовою аудиторією є викладачі, що потребують зручного інструменту для публікації контенту, та студенти, яким необхідне інтерактивне середовище для підтримки інтересу до навчання. Продукт належить до галузі освітніх технологій (EdTech) та реалізується у вигляді адаптивного вебзастосунку, доступного з будь-яких пристроїв.

A.1 Підстави до розробки

Фундаментом для початку робіт є офіційне завдання на виконання дипломного проєкту, затверджене 20 квітня 2026 року. Затверджена тема випускної кваліфікаційної роботи: «Програмна реалізація інтерактивної освітньої платформи».

A.2 Призначення розробки

Головне призначення системи полягає у вирішенні проблеми високого відсотка незавершених онлайн-курсів (низького Retention Rate). Платформа автоматизує процеси публікації навчальних матеріалів, контролю успішності та безперервного моніторингу активності користувачів. Завдяки впровадженню ігрових тригерів (нарахування балів досвіду, система рівнів, розблокування досягнень) рутинний процес засвоєння інформації перетворюється на захопливий досвід, що стимулює систематичне навчання.

A.3 Вимоги до програми чи програмного виробу

A.3.1 Вимоги до функціональних характеристик

Вебсистема «EduSteam» повинна забезпечувати виконання наступних ключових функцій:

- чітке розмежування прав доступу на основі рольової моделі (Студент, Викладач, Адміністратор);
- створення, редагування та структурування навчальних курсів (додавання модулів, текстових лекцій та мультимедіа) для викладачів;
- формування практичних завдань (квестів) із можливістю призначення конкретної винагороди за їх проходження;
- автоматичний фоновий трекінг часу, який користувач витрачає на вивчення матеріалів;
- динамічне нарахування очок досвіду (XP) та підвищення рівня профілю на основі навчальної активності;
- автоматизовану видачу віртуальних винагород (Achievements) при виконанні заданих умов.

Вхідною інформацією є облікові дані користувачів, текстовий та медіаконтент курсів, а також дії студентів (кліки, час сесії). Вихідною інформацією є оновлені показники прогресу на дашборді, сформовані каталоги курсів та статистичні звіти.

A.3.2 Вимоги до надійності

Для гарантії безперебійної роботи застосовано архітектурний підхід Server-Side Rendering (SSR) на базі фреймворку Next.js, що забезпечує швидке завантаження сторінок навіть при повільному інтернет-з'єднанні. Захист персональної інформації реалізується через збереження токенів авторизації виключно у HTTP-only cookies та криптографічне хешування паролів за стандартом bcrypt. На рівні бази даних (PostgreSQL) надійність

забезпечується строгим дотриманням принципів ACID та використанням зовнішніх ключів із каскадним видаленням для уникнення аномалій у даних.

A.3.3 Умови експлуатації

Взаємодія з програмним продуктом відбувається виключно через браузер (наприклад, Google Chrome, Microsoft Edge, Safari). Додаткового програмного забезпечення встановлювати не потрібно. Графічний інтерфейс має бути адаптивним (Responsive), підтримуючи коректне відображення на моніторах ПК, планшетах та смартфонах. Дизайн проєктується з урахуванням сучасних ергономічних стандартів для зниження когнітивного навантаження під час навчання.

A.3.4 Вимоги до складу параметрів технічних засобів

Для комфортної роботи з вебплатформою апаратне забезпечення клієнта має відповідати таким базовим критеріям: – операційна система: будь-яка сучасна десктопна (Windows, macOS, Linux) або мобільна (iOS, Android); – процесор: з тактовою частотою від 1.5 ГГц; – оперативна пам'ять: від 2 Гб для плавної роботи клієнтських скриптів; – стабільне підключення до мережі Інтернет (рекомендована швидкість від 2 Мбіт/с).

A.3.5 Вимоги до інформаційної й програмної сумісності

Внутрішня логіка та інтерфейс розробляються мовою суворої типізації TypeScript. Екосистема проєкту включає: – фронтенд та бекенд: Next.js (на базі бібліотеки React); – взаємодія з БД: об'єктно-реляційний перетворювач Prisma ORM; – база даних: реляційна СУБД PostgreSQL. Обмін даними між клієнтом і сервером оптимізовано за допомогою механізму Server Actions.

A.3.6 Вимоги до маркування й упакування

Фізичне пакування та маркування для даного продукту не передбачено, оскільки він розповсюджується виключно в електронному вигляді. Доступ до платформи надається у форматі "Програмне забезпечення як послуга" (SaaS). Водночас програмний код адаптований для контейнеризації, що дозволяє корпоративним клієнтам розгортати систему на власних потужностях (On-Premise) без необхідності отримання фізичних носіїв.

A.3.7 Вимоги до транспортування й збереження

Оскільки платформа «EduSteam» є вебзастосунком, вона не підлягає фізичному транспортуванню чи складському зберіганню. Робочі бази даних зберігаються на захищених хмарних серверах із налаштованим автоматичним резервним копіюванням (бекапами). Оновлення бізнес-логіки чи розширення каталогу ігрових досягнень здійснюється віддалено через системи безперервної інтеграції (CI/CD).

A.4 Вимоги до програмної документації

Супровідна документація складається з цього Технічного завдання та основного тексту пояснювальної записки дипломного проєкту. У документах деталізовано архітектурні рішення, наведено схеми реляційної бази даних, розкрито алгоритми роботи гейміфікаційного ядра та описано методику розгортання проєкту в середовищі розробника.

A.5 Техніко–економічні показники

Очікується, що впровадження платформи «EduSteam» дозволить освітнім установам зменшити показник відсіву студентів (Drop-out rate) щонайменше на 20-30% завдяки утриманню уваги через ігрову мотивацію. Економічний ефект досягається за рахунок збереження клієнтської бази для платних курсів та зниження витрат на адміністративний супровід процесу навчання.

A.6 Стадії й етапи розробки

Життєвий цикл розробки поділено на такі послідовні кроки:

- збір вимог, огляд існуючих систем та проєктування загальної концепції (1 тиждень);
- побудова ER-діаграми та налаштування бази даних PostgreSQL (1 тиждень);
- програмування бекенду, налаштування маршрутизації та безпеки (2 тижні);
- створення клієнтського інтерфейсу та ігрового функціоналу (2 тижні);
- фінальне тестування, усунення багів та написання звітної документації (1 тиждень).

A.7 Порядок контролю та приймання

Готовність системи оцінюється на основі проходження серії тестувань. Функціональне тестування підтверджує правильність роботи реєстрації, створення курсів та коректність перерахунку балів досвіду. Навантажувальне тестування перевіряє стійкість системи до одночасного доступу багатьох користувачів. Тестування безпеки гарантує неможливість

несанкціонованої зміни ролей або накрутки ігрового прогресу. Після успішного завершення всіх перевірок продукт вважається повністю готовим до інтеграції у реальний навчальний процес.

ДОДАТОК Б
Текст програми

Б.1 Структура бази даних (Prisma Schema)

```
// prisma/schema.prisma

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

// 1. Довідник категорій курсів
model Category {
  id      Int      @id @default(autoincrement())
  name    String   @unique // Наприклад: "Програмування", "Дизайн"
  courses Course[]
}

// 2. Довідник складності курсів
model Difficulty {
  id      Int      @id @default(autoincrement())
  level   String   @unique // Наприклад: "Новачок", "Спеціаліст", "Майстер"
  courses Course[]
}

// 3. Таблиця користувачів (Студенти)
model User {
  id          Int      @id @default(autoincrement())
  email       String   @unique
  first_name  String
  last_name   String
  password    String
  xp          Int      @default(0)
  level       Int      @default(1)
  created_at  DateTime @default(now())

  profile          Profile?
  enrollments      Enrollment[]
  achievements      UserAchievement[]
  quests           UserQuest[]
  posts            Post[]
  comments          Comment[]
  quest_comments   QuestComment[]

  sent_requests    FriendRequest[] @relation("SentRequests")
  received_requests FriendRequest[] @relation("ReceivedRequests")
  friendships1     Friendship[]    @relation("UserFriendships1")
  friendships2     Friendship[]    @relation("UserFriendships2")
  created_groups   CommunityGroup[]
  group_memberships CommunityGroupMember[]
  group_join_requests GroupJoinRequest[]

  sent_messages    Message[] @relation("SentMessages")
  received_messages Message[] @relation("ReceivedMessages")
}

// Додатковий профіль користувача
model Profile {
  id      Int      @id @default(autoincrement())
  bio     String?  // Про себе
  avatar_url String? // Посилання на аватар
  github_link String? // Посилання на соціальні мережі
  linkedin String?
  country  String?
  visibility String @default("public") // "public", "friends", "private"

  user_id Int @unique
  user    User @relation(fields: [user_id], references: [id])
}

// 4. Таблиця викладачів
model Teacher {
  id      Int      @id @default(autoincrement())
  first_name String
  last_name String
}
```

```

email          String    @unique
password       String
specialization String?
bio            String?
is_approved    Boolean    @default(false)
created_at     DateTime   @default(now())

courses Course[]

// Зв'язки для системи друзів та чатів з викладачами
sent_requests  FriendRequest[] @relation("TeacherSentRequests")
received_requests FriendRequest[] @relation("TeacherReceivedRequests")
friendships1   Friendship[]    @relation("TeacherFriendships1")
friendships2   Friendship[]    @relation("TeacherFriendships2")
sent_messages  Message[]       @relation("TeacherSentMessages")
received_messages Message[]    @relation("TeacherReceivedMessages")
}

// 5. Таблиця навчальних курсів
model Course {
  id          Int        @id @default(autoincrement())
  title       String
  description  String?
  is_published Boolean    @default(false)
  created_at  DateTime   @default(now())

  teacher_id Int
  teacher     Teacher @relation(fields: [teacher_id], references: [id])

  category_id Int
  category     Category @relation(fields: [category_id], references: [id])
  difficulty_id Int
  difficulty    Difficulty @relation(fields: [difficulty_id], references: [id])

  enrollments Enrollment[]
  quests       Quest[]
  achievements Achievement[]
}

// 6. Зв'язок Студент + Курс (трекер часу)
model Enrollment {
  id          Int        @id @default(autoincrement())
  user_id     Int
  course_id   Int
  total_minutes Int      @default(0)
  enrolled_at DateTime   @default(now())

  user  User @relation(fields: [user_id], references: [id])
  course Course @relation(fields: [course_id], references: [id])

  @@unique([user_id, course_id])
}

// 7. Квести (Завдання) всередині курсу
model Quest {
  id          Int        @id @default(autoincrement())
  title       String
  description  String?
  xp_reward   Int
  order       Int        @default(0)
  course_id   Int
  course      Course @relation(fields: [course_id], references: [id])

  users      UserQuest[]
  quest_comments QuestComment[]
  attachments QuestAttachment[]
}

// 8. Зв'язок виконаних квестів користувачами
model UserQuest {
  id          Int        @id @default(autoincrement())
  user_id     Int
  quest_id    Int
  completed_at DateTime   @default(now())

  user  User @relation(fields: [user_id], references: [id])
  quest Quest @relation(fields: [quest_id], references: [id])

  @@unique([user_id, quest_id])
}

```

```

// 9. Досягнення (Гейміфікаційні ачивки)
model Achievement {
  id          Int          @id @default(autoincrement())
  name        String
  description  String
  icon        String?

  course_id Int?
  course     Course? @relation(fields: [course_id], references: [id])

  users UserAchievement[]
}

// 10. Таблиця отриманих користувачами досягнень
model UserAchievement {
  id          Int          @id @default(autoincrement())
  user_id     Int
  achievement_id Int
  earned_at   DateTime @default(now())

  user        User          @relation(fields: [user_id], references: [id])
  achievement Achievement @relation(fields: [achievement_id], references: [id])

  @@unique([user_id, achievement_id])
}

// 11. Дописи в спільноті
model Post {
  id          Int          @id @default(autoincrement())
  content     String
  media_url   String?
  created_at  DateTime @default(now())
  is_edited   Boolean @default(false)

  user_id Int
  user      User @relation(fields: [user_id], references: [id])

  group_id Int?
  group     CommunityGroup? @relation(fields: [group_id], references: [id], onDelete: Cascade)

  comments Comment[]
}

// Коментарі до дописів
model Comment {
  id          Int          @id @default(autoincrement())
  content     String
  created_at  DateTime @default(now())

  user_id Int
  post_id Int

  user User @relation(fields: [user_id], references: [id])
  post Post @relation(fields: [post_id], references: [id], onDelete: Cascade)
}

// 12. Вайтліст дозволених викладачів
model AllowedTeacher {
  id          Int          @id @default(autoincrement())
  email       String       @unique
  full_name   String?
  notes       String?
  created_at  DateTime @default(now())
}

// 13. Заявки на додавання у друзі
model FriendRequest {
  id          Int          @id @default(autoincrement())
  sender_id   Int?
  receiver_id Int?
  teacher_sender_id Int?
  teacher_receiver_id Int?
  status      String       @default("PENDING") // PENDING, ACCEPTED, REJECTED
  created_at  DateTime @default(now())

  sender      User? @relation("SentRequests", fields: [sender_id], references: [id])
  receiver    User? @relation("ReceivedRequests", fields: [receiver_id], references: [id])
}

```

```

    teacher_sender Teacher? @relation("TeacherSentRequests", fields: [teacher_sender_id],
references: [id])
    teacher_receiver Teacher? @relation("TeacherReceivedRequests", fields:
[teacher_receiver_id], references: [id])
}

// 14. Підтверджені дружні зв'язки (Користувач-Користувач, Користувач-Викладач)
model Friendship {
    id Int @id @default(autoincrement())
    user1_id Int?
    user2_id Int?
    teacher1_id Int?
    teacher2_id Int?
    created_at DateTime @default(now())

    user1 User? @relation("UserFriendships1", fields: [user1_id], references: [id])
    user2 User? @relation("UserFriendships2", fields: [user2_id], references: [id])
    teacher1 Teacher? @relation("TeacherFriendships1", fields: [teacher1_id], references: [id])
    teacher2 Teacher? @relation("TeacherFriendships2", fields: [teacher2_id], references: [id])
}

// 15. Групи спільноти
model CommunityGroup {
    id Int @id @default(autoincrement())
    name String
    description String?
    avatar_url String? // Аватарка групи
    creator_id Int
    is_private Boolean @default(false)
    created_at DateTime @default(now())

    creator User @relation(fields: [creator_id], references: [id])
    members CommunityGroupMember[]
    posts Post[]
    join_requests GroupJoinRequest[]
}

// Заявки на вступ у закриті групи спільноти
model GroupJoinRequest {
    id Int @id @default(autoincrement())
    group_id Int
    user_id Int
    status String @default("PENDING") // PENDING, APPROVED, REJECTED
    created_at DateTime @default(now())

    group CommunityGroup @relation(fields: [group_id], references: [id], onDelete: Cascade)
    user User @relation(fields: [user_id], references: [id], onDelete: Cascade)

    @@unique([group_id, user_id])
}

// 16. Члени груп
model CommunityGroupMember {
    group_id Int
    user_id Int
    joined_at DateTime @default(now())

    group CommunityGroup @relation(fields: [group_id], references: [id], onDelete: Cascade)
    user User @relation(fields: [user_id], references: [id], onDelete: Cascade)

    @@id([group_id, user_id])
}

// 17. Приватні повідомлення (Чати між друзями)
model Message {
    id Int @id @default(autoincrement())
    content String
    created_at DateTime @default(now())
    is_read Boolean @default(false)

    sender_id Int?
    receiver_id Int?
    teacher_sender_id Int?
    teacher_receiver_id Int?

    sender User? @relation("SentMessages", fields: [sender_id], references: [id])
    receiver User? @relation("ReceivedMessages", fields: [receiver_id], references:
[id])
    teacher_sender Teacher? @relation("TeacherSentMessages", fields: [teacher_sender_id],
references: [id])
}

```

```

    teacher_receiver Teacher? @relation("TeacherReceivedMessages", fields:
[teacher_receiver_id], references: [id])
}

// 18. Коментарі до квестів
model QuestComment {
  id          Int          @id @default(autoincrement())
  content     String
  created_at  DateTime @default(now())

  user_id Int
  quest_id Int

  user User @relation(fields: [user_id], references: [id])
  quest Quest @relation(fields: [quest_id], references: [id], onDelete: Cascade)
}

// 19. Вкладені файли до квестів
model QuestAttachment {
  id          Int          @id @default(autoincrement())
  file_url    String
  file_name   String
  created_at  DateTime @default(now())

  user_id Int
  quest_id Int

  quest Quest @relation(fields: [quest_id], references: [id], onDelete: Cascade)
}

```

Б.2 Логіка гейміфікації (Нарахування XP та Рівнів)

Б.2.1 Утиліта перевірки умов та видачі досягнень

(app/lib/achievements.ts)

```

// app/lib/achievements.ts
import { PrismaClient } from "@prisma/client";

type AchievementResult = {
  id: number;
  name: string;
  description: string;
  icon: string | null;
};

/**
 * Перевіряє умови досягнень і нараховує нові.
 * Повертає масив щойно отриманих досягнень (для динамічного показу тоста).
 */
export async function checkAndAwardAchievements(
  userId: number,
  prisma: PrismaClient
): Promise<AchievementResult[]> {
  // Завантажуємо повні дані користувача одним оптимізованим запитом
  const user = await prisma.user.findUnique({
    where: { id: userId },
    include: {
      achievements: { select: { achievement_id: true } },
      enrollments: { select: { id: true, total_minutes: true } },
      quests: { select: { id: true } },
      posts: { select: { id: true } },
      friendships1: { select: { id: true } },
      friendships2: { select: { id: true } },
      group_memberships: { select: { group_id: true } },
    },
  });
  if (!user) return [];

  const earnedIds = new Set(user.achievements.map((a) => a.achievement_id));
  const allAchievements = await prisma.achievement.findMany({
    where: { course_id: null }, // тільки глобальні ачивки системи
    orderBy: { id: "asc" },
  });
}

```

```

const totalMinutes = user.enrollments.reduce((sum, e) => sum + e.total_minutes, 0);
const friendsCount = user.friendships1.length + user.friendships2.length;

// Визначення умов для кожного досягнення за назвою
const conditions: Record<string, boolean> = {
  "Перший крок": user.enrollments.length >= 1,
  "Перша перемога": user.quests.length >= 1,
  "Новачок": user.level >= 2,
  "Допитливий розум": user.enrollments.length >= 3,
  "Колекціонер знань": user.enrollments.length >= 5,
  "Марафонець": totalMinutes >= 60,
  "Невтомний учень": totalMinutes >= 600, // 10 годин навчання
  "ХР Гравець": user.xp >= 100,
  "Досвідчений": user.xp >= 500,
  "Майстер EduSteam": user.xp >= 1000,
  "Легенда": user.level >= 10,
  "Знайомий": friendsCount >= 1,
  "Соціальний метелик": friendsCount >= 5,
  "Перший пост": user.posts.length >= 1,
  "Активний член": user.group_memberships.length >= 1,
};

const newlyEarned: AchievementResult[] = [];

for (const achievement of allAchievements) {
  if (earnedIds.has(achievement.id)) continue;
  const qualified = conditions[achievement.name];
  if (!qualified) continue;

  try {
    // Зберігаємо нове досягнення в БД
    await prisma.userAchievement.create({
      data: { user_id: userId, achievement_id: achievement.id },
    });
    newlyEarned.push({
      id: achievement.id,
      name: achievement.name,
      description: achievement.description,
      icon: achievement.icon,
    });
  } catch {
    // Запобігання конфліктам у разі паралельного запиту
  }
}

return newlyEarned;
}

```

Б.2.2 API маршрут виконання квестів з формулою розрахунку рівня (app/api/quests/[id]/complete/route.ts)

```

// app/api/quests/[id]/complete/route.ts
import { NextRequest, NextResponse } from "next/server";
import { cookies } from "next/headers";
import { verifyToken } from "@app/lib/auth";
import { prisma } from "@app/lib/prisma";
import { getSession } from "@app/lib/session";
import { checkAndAwardAchievements } from "@app/lib/achievements";

// POST – виконати квест та нарахувати досвід
export async function POST(
  req: Request,
  { params }: { params: Promise<{ id: string }> }
) {
  try {
    const cookieStore = await cookies();
    const token = cookieStore.get("token")?.value;
    if (!token) return NextResponse.json({ error: "Не авторизований" }, { status: 401 });

    const user = await verifyToken(token);
    if (!user) return NextResponse.json({ error: "Невалідний токен" }, { status: 401 });

```

```

const { id } = await params;
const questId = parseInt(id);

const quest = await prisma.quest.findUnique({ where: { id: questId }, include: { course:
true } });
if (!quest) return NextResponse.json({ error: "Квест не знайдено" }, { status: 404 });

const enrollment = await prisma.enrollment.findUnique({
  where: { user_id_course_id: { user_id: user.id, course_id: quest.course_id } },
});
if (!enrollment) return NextResponse.json({ error: "Спочатку запишись на курс" }, {
status: 403 });

const alreadyDone = await prisma.userQuest.findUnique({
  where: { user_id_quest_id: { user_id: user.id, quest_id: questId } },
});
if (alreadyDone) return NextResponse.json({ error: "Квест вже виконаний" }, { status: 400
});

// Використання ACID-транзакції для безпечного нарахування XP та Level
const result = await prisma.$transaction(async (tx) => {
  await tx.userQuest.create({ data: { user_id: user.id, quest_id: questId } });

  const currentUser = await tx.user.findUnique({
    where: { id: user.id },
    select: { xp: true, level: true },
  });
  if (!currentUser) throw new Error("Користувача не знайдено");

  const newXp = currentUser.xp + quest.xp_reward;
  let newLevel = currentUser.level;

  // Математична формула розрахунку рівнів: Рівень = sqrt(XP / 100)
  while (newXp >= newLevel * newLevel * 100) {
    newLevel++;
  }

  const updatedUser = await tx.user.update({
    where: { id: user.id },
    data: { xp: newXp, level: newLevel },
  });

  return {
    xp_gained: quest.xp_reward,
    new_xp: updatedUser.xp,
    new_level: updatedUser.level,
    leveled_up: newLevel > currentUser.level,
  };
});

return NextResponse.json({
  message: "Квест виконано!",
  ...result,
  earned_achievements: await checkAndAwardAchievements(user.id, prisma)
}, { status: 200 });

} catch (error) {
  console.error("Помилка при виконанні квесту:", error);
  return NextResponse.json({ error: "Внутрішня помилка сервера" }, { status: 500 });
}
}

// DELETE – скасувати виконання квесту (корекція XP та рівнів у зворотний бік)
export async function DELETE(
  req: NextRequest,
  { params }: { params: Promise<{ id: string }> }
) {
  const session = await getSession();
  if (!session) return NextResponse.json({ error: "Unauthorized" }, { status: 401 });

  const { id } = await params;
  const questId = parseInt(id);
  if (isNaN(questId)) return NextResponse.json({ error: "Invalid ID" }, { status: 400 });

  try {
    const quest = await prisma.quest.findUnique({ where: { id: questId } });
    if (!quest) return NextResponse.json({ error: "Quest not found" }, { status: 404 });

    const userQuest = await prisma.userQuest.findUnique({
      where: { user_id_quest_id: { user_id: session.id, quest_id: questId } },

```

```

    });
    if (!userQuest) return NextResponse.json({ error: "Quest not completed" }, { status: 404 });
  });

  await prisma.$transaction(async (tx) => {
    await tx.userQuest.delete({
      where: { user_id_quest_id: { user_id: session.id, quest_id: questId } },
    });

    const currentUser = await tx.user.findUnique({
      where: { id: session.id },
      select: { xp: true, level: true },
    });
    if (!currentUser) return;

    const newXp = Math.max(0, currentUser.xp - quest.xp_reward);
    let newLevel = currentUser.level;
    while (newLevel > 1 && newXp < (newLevel - 1) * (newLevel - 1) * 100) {
      newLevel--;
    }

    await tx.user.update({
      where: { id: session.id },
      data: { xp: newXp, level: newLevel },
    });
  });

  return NextResponse.json({ success: true, xp_removed: quest.xp_reward });
} catch (error) {
  console.error("Undo quest error:", error);
  return NextResponse.json({ error: "Server error" }, { status: 500 });
}
}

```

Б.3 Механізм безпеки та авторизації (Middleware / Auth)

Б.3.1 Криптографічні хелпери авторизації (app/lib/auth.ts)

```

// app/lib/auth.ts
import { SignJWT, jwtVerify } from 'jose'

// Секретний ключ для шифрування підпису
const secret = new TextEncoder().encode(
  process.env.JWT_SECRET || 'edusteam_secret_key_change_in_production'
)

type TokenPayload = {
  id: number
  email: string
  role?: "student" | "teacher"
}

// Генерація токена з терміном дії 7 днів
export async function signToken(payload: TokenPayload) {
  const token = await new SignJWT(payload)
    .setProtectedHeader({ alg: 'HS256' })
    .setExpirationTime('7d')
    .sign(secret)

  return token
}

// Декодування та верифікація JWT токена
export async function verifyToken(token: string) {
  try {
    const { payload } = await jwtVerify(token, secret)
    return payload as unknown as TokenPayload
  } catch {
    return null
  }
}

```

Б.3.2 Хелпери перевірки сесії (app/lib/session.ts)

```
// app/lib/session.ts
import { cookies } from "next/headers";
import { verifyToken } from "../auth";

// Для студентів – зчитує куку "token"
export async function getSession() {
  const cookieStore = await cookies();
  const token = cookieStore.get("token")?.value;
  if (!token) return null;
  const payload = await verifyToken(token);
  if (!payload || !payload.id) return null;
  return payload;
}

// Для викладачів – зчитує куку "teacher_token"
export async function getTeacherSession() {
  const cookieStore = await cookies();
  const token = cookieStore.get("teacher_token")?.value;
  if (!token) return null;
  const payload = await verifyToken(token);
  if (!payload || !payload.id) return null;
  return payload;
}

// Для спільних сторінок – підтримує обидва типи профілів
export async function getAnySession(): Promise<{ id: number; role: "student" | "teacher" } | null> {
  const cookieStore = await cookies();

  // 1. Спочатку перевіряємо токен викладача
  const teacherToken = cookieStore.get("teacher_token")?.value;
  if (teacherToken) {
    const payload = await verifyToken(teacherToken);
    if (payload && payload.id) return { id: Number(payload.id), role: "teacher" };
  }

  // 2. Потім перевіряємо токен студента
  const studentToken = cookieStore.get("token")?.value;
  if (studentToken) {
    const payload = await verifyToken(studentToken);
    const validId = payload?.id || (payload as any)?.userId;
    if (validId) return { id: Number(validId), role: "student" };
  }

  return null;
}
```

Б.3.3 Глобальний фільтр безпеки маршрутів (proxy.ts)

```
// proxy.ts
import { NextResponse } from "next/server";
import type { NextRequest } from "next/server";
import { verifyToken } from "../app/lib/auth";

// Маршрути з роздільним доступом
const protectedRoutes = ["/dashboard", "/home", "/community", "/chat", "/friends",
"/settings", "/courses/create", "/achievements"];
const teacherRoutes = ["/teacher/home", "/teacher/courses", "/teacher/groups",
"/teacher/profile", "/teacher/settings"];

export async function proxy(request: NextRequest) {
  const token = request.cookies.get("token")?.value;
  const pathname = request.nextUrl.pathname;

  const user = token ? await verifyToken(token) : null;
  const teacherToken = request.cookies.get("teacher_token")?.value;
  const teacherUser = teacherToken ? await verifyToken(teacherToken) : null;

  const isProtected = protectedRoutes.some((route) => pathname.startsWith(route));
```

```

// Заборона неавторизованого доступу до студентських сторінок
if (isProtected && !user && !teacherUser) {
  return NextResponse.redirect(new URL("/login", request.url));
}

const isAuthRoute = pathname === "/" || pathname.startsWith("/login") ||
pathname.startsWith("/register");
if (isAuthRoute && user) {
  return NextResponse.redirect(new URL("/home", request.url));
}

// Захист спеціалізованих кабінетів викладача
const isTeacherRoute = teacherRoutes.some((route) => pathname.startsWith(route));
if (isTeacherRoute) {
  if (!teacherUser || teacherUser.role !== "teacher") {
    return NextResponse.redirect(new URL("/teacher/login", request.url));
  }
}

// Захист адміністративної панелі (перевірка суперключа)
const isAdminRoute = pathname.startsWith("/admin") && !pathname.startsWith("/admin/login");
if (isAdminRoute) {
  const adminSession = request.cookies.get("admin_session")?.value;
  const adminKey = process.env.ADMIN_SECRET_KEY || "edusteam_admin_2024";
  if (adminSession !== adminKey) {
    return NextResponse.redirect(new URL("/admin/login", request.url));
  }
}

return NextResponse.next();
}

export const config = {
  matcher: [
    // Виключає статичні ресурси та системні API з перехоплення
    "/((?!api|_next/static|_next/image|favicon.ico).*)",
  ],
};

```

Б.4 Клієнтський інтерактивний компонент (React / Інтерфейс)

```

// app/components/AchievementToast.tsx
"use client";

import { useEffect, useState } from "react";

type Achievement = {
  id: number;
  name: string;
  description: string;
  icon: string | null;
};

type Toast = Achievement & { toastId: number };

const SEEN_KEY = "edusteam_seen_achievement_ids";
let toastCounter = 0;

// Отримання списку вже показаних досягнень з LocalStorage
function getSeenIds(): Set<number> {
  try {
    const raw = localStorage.getItem(SEEN_KEY);
    return raw ? new Set(JSON.parse(raw) as number[]) : new Set();
  } catch {
    return new Set();
  }
}

// Запис показаних досягнень в LocalStorage, щоб не показувати повторно
function markSeen(ids: number[]) {
  try {
    const existing = getSeenIds();
    ids.forEach((id) => existing.add(id));
    localStorage.setItem(SEEN_KEY, JSON.stringify([...existing]));
  } catch {}
}

```

```

}

function addToast(achievement: Achievement, setToasts:
React.Dispatch<React.SetStateAction<Toast[]>>) {
  const toastId = ++toastCounter;
  setToasts((prev) => [...prev, { ...achievement, toastId }]);
  // Автоматичне зникнення через 6 секунд
  setTimeout(() => {
    setToasts((prev) => prev.filter((t) => t.toastId !== toastId));
  }, 6000);
}

export default function AchievementToast() {
  const [toasts, setToasts] = useState<Toast[]>([]);

  // 1. Опитування сервера при першому завантаженні (запобігання пропущеним ачивкам)
  useEffect(() => {
    const checkNewAchievements = async () => {
      try {
        const res = await fetch("/api/user/achievements");
        if (!res.ok) return;
        const achievements: Achievement[] = await res.json();
        if (!achievements.length) return;

        const seen = getSeenIds();
        const unseen = achievements.filter((a) => !seen.has(a.id));

        if (unseen.length === 0) return;

        // Почерговий показ тостів з невеликим інтервалом
        unseen.forEach((a, index) => {
          setTimeout(() => addToast(a, setToasts), index * 800);
        });

        markSeen(achievements.map((a) => a.id));
      } catch {
        // Заглушка помилки мережі
      }
    };

    const timer = setTimeout(checkNewAchievements, 1500);
    return () => clearTimeout(timer);
  }, []);

  // 2. Підписка на подію реального часу "achievement-earned"
  useEffect(() => {
    const achievementHandler = (e: Event) => {
      const achievement = (e as CustomEvent<Achievement>).detail;
      addToast(achievement, setToasts);
      markSeen([achievement.id]);
    };
    window.addEventListener("achievement-earned", achievementHandler);
    return () => window.removeEventListener("achievement-earned", achievementHandler);
  }, []);

  // 3. Підписка на подію додавання у друзі
  useEffect(() => {
    const friendHandler = (e: Event) => {
      const { name } = (e as CustomEvent<{ name: string }>).detail;
      addToast(
        {
          id: -1,
          name: `${name} тепер ваш друг!`,
          description: "Ви можете писати один одному у чаті",
          icon: "□"
        },
        setToasts
      );
    };
    window.addEventListener("friend-added", friendHandler);
    return () => window.removeEventListener("friend-added", friendHandler);
  }, []);

  if (toasts.length === 0) return null;

  return (
    <div className="fixed bottom-6 right-6 z-50 flex flex-col gap-3 pointer-events-none">
      {toasts.map((toast) => (
        <div
          key={toast.toastId}

```

```

        className="pointer-events-auto flex items-center gap-0 w-80 rounded-sm overflow-
hidden shadow-2xl"
        style={{
            animation: "slideInAchievement 0.4s cubic-bezier(0.175, 0.885, 0.32, 1.275)",
            background: "linear-gradient(135deg, #1b2838 0%, #2a475e 100%)",
            border: "1px solid #4c6b8c",
            boxShadow: "0 0 20px rgba(102,192,244,0.15), 0 8px 32px rgba(0,0,0,0.6)",
        }}
    >
    { /* Декоративна Steam-смуга ліворуч */ }
    <div
        style={{
            width: "4px",
            alignSelf: "stretch",
            background: "linear-gradient(180deg, #66c0f4, #1a9fff)",
            flexShrink: 0,
        }}
    />

    { /* Контейнер іконки досягнення */ }
    <div
        className="flex-shrink-0 w-14 h-14 flex items-center justify-center text-3xl ml-3"
        style={{
            background: "rgba(102,192,244,0.08)",
            border: "1px solid rgba(102,192,244,0.15)",
            borderRadius: "6px",
        }}
    >
        {toast.icon || "☐"}
    </div>

    { /* Текстовий опис сповіщення */ }
    <div className="flex-1 min-w-0 py-3 px-3">
        <p
            className="text-xs uppercase tracking-widest mb-0.5"
            style={{ color: "#66c0f4", fontWeight: 600, letterSpacing: "0.1em" }}
        >
            Досягнення отримано!
        </p>
        <p className="text-white font-semibold text-sm leading-tight">
            {toast.name}
        </p>
        {toast.description !== "???" && (
            <p className="text-xs mt-0.5 leading-snug line-clamp-2" style={{ color:
"#8f98a0" }}>
                {toast.description}
            </p>
        )}
    </div>

    { /* Кнопка ручного закриття */ }
    <button
        onClick={() => setToasts((prev) => prev.filter((t) => t.toastId !==
toast.toastId))}
        className="self-start mt-2 mr-2 text-gray-600 hover:text-gray-400 transition-
colors text-xl leading-none flex-shrink-0"
        aria-label="Закрити"
    >
        x
    </button>
</div>
)))

{ /* Вбудовані стилі ключових кадрів анімації */ }
<style jsx global>{`
    @keyframes slideInAchievement {
        from {
            opacity: 0;
            transform: translateX(100%) scale(0.85);
        }
        to {
            opacity: 1;
            transform: translateX(0) scale(1);
        }
    }
`}</style>
</div>
);
}

```

Б.5 Строга типізація (TypeScript Interfaces)

```
// Збірна специфікація типів TypeScript для EduSteam

// 1. Авторизаційні токени та права доступу (app/lib/auth.ts)
export type TokenPayload = {
  id: number;
  email: string;
  role?: "student" | "teacher";
};

// 2. Структури системи миттєвих повідомлень (app/chat/ChatClient.tsx)
export type Friend = {
  id: number;
  first_name: string;
  last_name: string;
  level: number;
  profile?: { avatar_url: string | null } | null;
  role?: string;
};

export type Contact = {
  friend: Friend;
  lastMessage: {
    content: string;
    created_at: string;
    sender_id: number;
  } | null;
  unreadCount: number;
};

export type Message = {
  id: number;
  content: string;
  created_at: string;
  sender_id: number;
  receiver_id: number;
};

// 3. Сутності стрічки новин та постів спільноти (app/community/CommunityFeed.tsx)
export type postData = {
  id: number;
  content: string;
  media_url: string | null;
  created_at: string;
  is_edited: boolean;
  user_id: number;
  user: {
    first_name: string;
    last_name: string;
    level: number;
    profile: { avatar_url: string | null } | null;
  };
  comments: CommentData[];
  _count?: {
    comments: number;
  };
};

export type CommentData = {
  id: number;
  content: string;
  created_at: string;
  user: {
    id: number;
    first_name: string;
    last_name: string;
    level: number;
    profile: { avatar_url: string | null } | null;
  };
};

// 4. Типи деталей квесту: коментарі та файли
(app/courses/[id]/quests/[questId]/QuestDetailClient.tsx)
export type CommentUser = {
  id: number;
  first_name: string;
```

```

    last_name: string;
    level: number;
    profile: { avatar_url: string | null } | null;
  };

export type QuestComment = {
  id: number;
  content: string;
  created_at: string;
  user: CommentUser;
};

export type QuestAttachment = {
  id: number;
  file_name: string;
  file_url: string;
  created_at: string;
};

// 5. Статистика проходження курсів для викладачів
(app/teacher/courses/[id]/CourseStatistics.tsx)
export type StudentStat = {
  id: number;
  first_name: string;
  last_name: string;
  avatar_url: string | null;
  completedQuests: number;
  totalQuests: number;
  totalMinutes: number;
  enrolledAt: string;
};

export type StatisticsData = {
  courseTitle: string;
  totalStudents: number;
  completedStudents: number;
  averageProgress: number;
  students: StudentStat[];
};

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Тема: «Програмна реалізація інтерактивної освітньої платформи»

Виконав:
студент групи КНТ-112

Микита ПАСТУШЕНКО

Керівник:
к.т.н., доцент

Євгеній ГОФМАН

Рисунок В.1 – Слайд 1

Мета, об'єкт та предмет дослідження

•**Мета:** Підвищення рівня залученості користувачів та зниження трудомісткості адміністрування навчального процесу за рахунок програмної реалізації інтерактивної вебплатформи з автоматизованим відстеженням прогресу, системою квестів та ігрових досягнень.

•**Об'єкт дослідження:** Процес інженерії програмного забезпечення для дистанційного навчання з використанням мотиваційних механік.

•**Предмет дослідження:** Методи та програмні інструменти для побудови освітніх вебплатформ з інтегрованою системою гейміфікації.

2

Рисунок В.2 – Слайд 2

Аналіз існуючих рішень

Критерій порівняння	Coursera	Duolingo	Moodle	EduSteam (Розробка)
Основний фокус платформи	Масові лекції	Мікронавчання	Адміністрування	Інтерактивні курси
Ступінь гейміфікації	Низький	Дуже високий	Базовий (через плагіни)	Високий (XP, досягнення)
Створення кастомних курсів викладачами	Лише для партнерів	Неможливо	Так (але складно)	Так (інтуїтивно)
Створення кастомних Автоматичний трекінг часу навчання	Частково	Ні	Так	Так (у реальному часі)
Архітектура клієнтської частини	SPA (React)	SPA / Native	Monolith (PHP)	SSR (Next.js)
Сучасний UI/UX дизайн	Добре	Відмінно	Незадовільно	Відмінно
Складність розгортання системи	Хмарна (SaaS)	Хмарна (SaaS)	Висока (Self-hosted)	Низька (Docker/Vercel)

Порівняльний аналіз популярних LMS та освітніх платформ (Moodle, Coursera, Duolingo) виявив дефіцит комплексних рішень, які поєднують публікацію контенту з глибокою гейміфікацією.

3

Рисунок В.3 – Слайд 3

Вибір технологічного стеку

Критерій порівняння	JavaScript	Python	TypeScript
Типізація	Динамічна	Динамічна	Суворі статична
Виявлення помилок	Під час виконання	Під час виконання	На етапі компіляції
Масштабованість	Середня	Висока	Дуже висока
Інтеграція з React/Next.js	Рідна	Складна	Відмінна (повна підтримка)

4

Рисунок В.4 – Слайд 4

Структурна схема підсистем



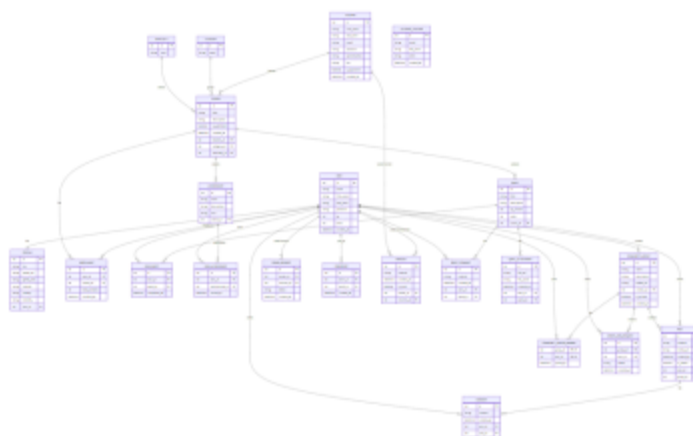
Архітектуру платформи «EduSteam» декомпозовано на 4 незалежні підсистеми:

- 1) Адміністрування профілів та користувачів.
- 2) Редагування навчального контенту.
- 3) Гейміфікація та нарахування досягнень.
- 4) Соціальна взаємодія спільноти.

5

Рисунок В.5 – Слайд 5

Проектування реляційної бази даних



База даних налічує 22 взаємопов'язані таблиці.

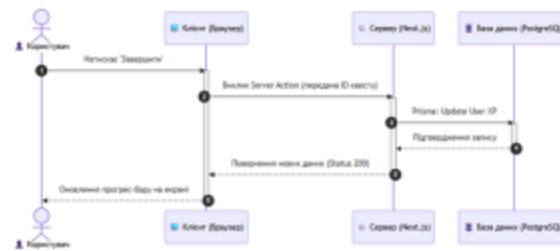
Забезпечено строгі реляційну цілісність (Data Integrity) та зв'язки 1:1, 1:N, M:N.

Налаштовано каскадне видалення (onDelete: Cascade) для запобігання аномаліям даних.

6

Рисунок В.6 – Слайд 6

Логіка системи гейміфікації



Тригер: Успішне завершення практичного квесту чи лекції.

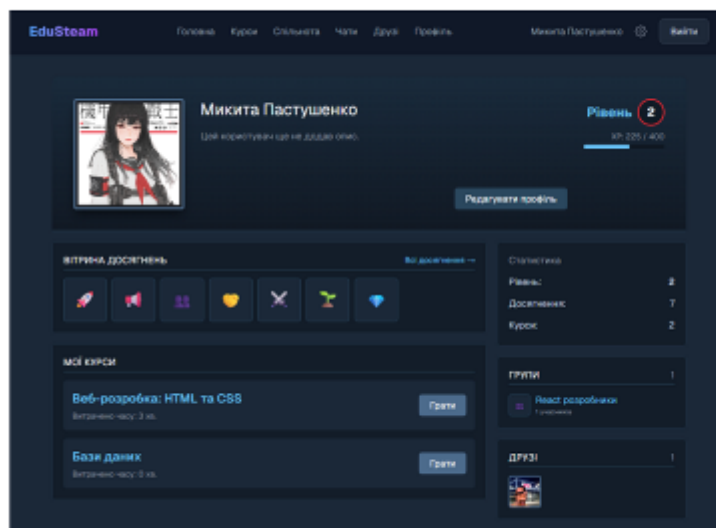
Обробка: Виклик Server Action для валідації результатів.

Нагорода: Автоматичне нарахування балів досвіду (XP) та перевірка умов для підвищення рівня.

7

Рисунок В.7 – Слайд 7

Клієнтський інтерфейс: Дашборд



Персоналізований робочий простір студента.

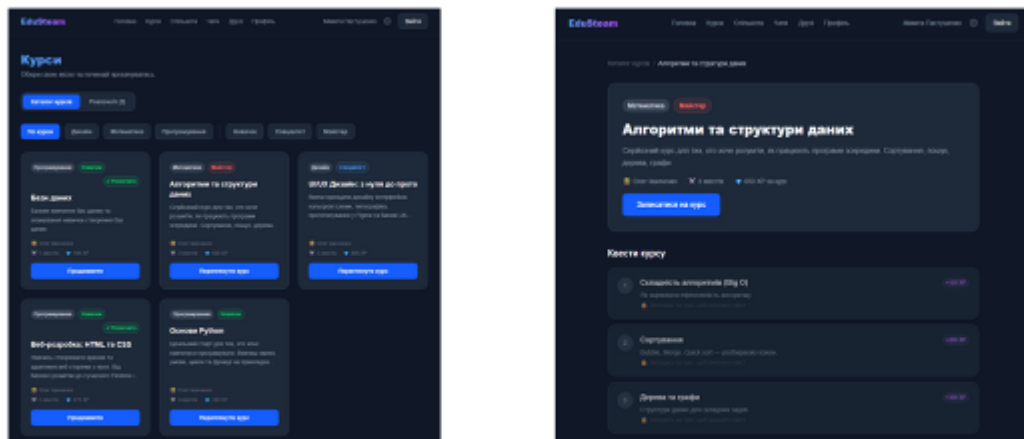
Інтерактивний прогрес-бар рівня та кількості XP.

Відстеження пройдених та активних курсів.

8

Рисунок В.8 – Слайд 8

Клієнтський інтерфейс: Навчальний контент



Модуль проходження лекцій та виконання квестів.

Адаптивний дизайн (Responsive UI) для мобільних та десктопних пристроїв.

Миттєвий зворотний зв'язок без перезавантаження сторінки завдяки React 19.

9

Рисунок В.9 – Слайд 9

Безпека та захист даних



Автентифікація: реалізована на базі JWT-токенів (бібліотека [jose](#)).

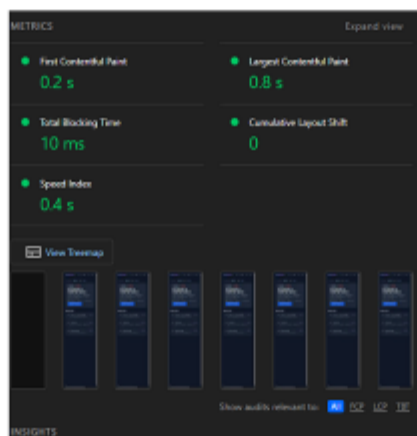
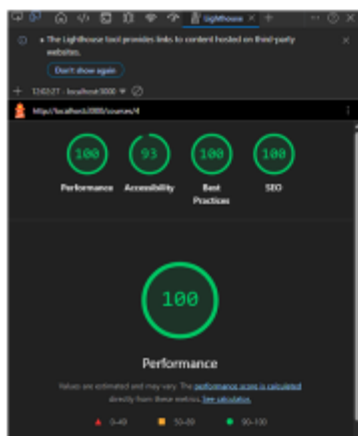
Захист сесій: збереження токенів виключно у HTTP-only cookies.

Криптографія: одностороннє хешування паролів за стандартом [bcryptjs](#).

10

Рисунок В.10 – Слайд 10

Експериментальне тестування



Функціональне тестування: перевірка коректності роботи модулів авторизації, створення курсів та підрахунку XP.

Продуктивність: застосування SSR забезпечило високу швидкість відмальовування інтерфейсу (LCP < 1.5 сек).

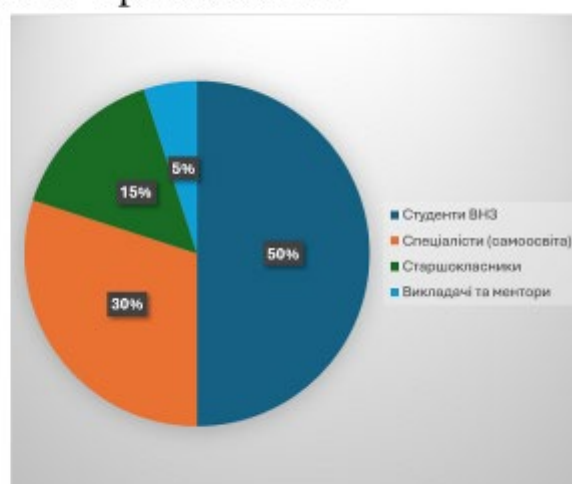
11

Рисунок В.11 – Слайд 11

Цільова аудиторія та ефективність

Економічний ефект:

- Зниження відсотка відсіву студентів (Drop-out rate).
- Економія часу викладачів на перевірку завдань.
- Можливість продажу преміум-курсів або підписки за моделлю SaaS.



12

Рисунок В.12 – Слайд 12

Загальні висновки та подальший розвиток

•Отримані результати:

- Спроектовано стабільну клієнт-серверну архітектуру освітньої платформи.
- Успішно інтегровано систему мотиваційних ігрових механік (XP, рівні, досягнення).
- Забезпечено високий рівень безпеки та захисту персональних даних.

•Вектори розвитку:

- Розробка PWA-версії (Progressive Web App) для мобільних пристроїв.
- Інтеграція з відеохостингами та модулями аналітики на базі штучного інтелекту.
- Впровадження спільнот (форумів) для прямої взаємодії студентів.