

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
І СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка інтерактивного веб-середовища освітнього ІТ порталу

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБ-22-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за
галузями)»


(підпис)

Павлушенко Дмитро
Миколайович
(прізвище, ім'я, по батькові)

Науковий
керівник


(підпис)

Редич Олександр
Володимирович канд. екон.
наук, доцент
(прізвище, ім'я, по батькові, науковий
ступінь, вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Ніжегородцев Владислав
Олександрович, канд. пед.
наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та
інформаційних технологій і систем протокол від « 12 » травня 2026 № 20

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, канд. фіз.-мат.
наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Ірпінь – 2026

Реєстрація № 57

28.05.2026

(дата)



(підпис відповідальної особи)

Результати перевірки

До захисту

(до захисту, на доопрацювання)

12.05.26

(дата)



(підпис)

Результат захисту

5 (відмінно)

(оцінка)

08.06.2026

(дата)



(підпис)

Голова комісії:



(підпис)

О.С. Дюбанов

Члени комісії:



(підпис)

М.М. Філоненко



(підпис)

В.О. Ніжегородцев



(підпис)

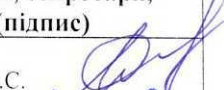
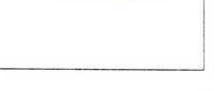
А.А. Омельчук

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії/членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	29		08.06.2026 Редич О. В.  Дюбанов О.С. 
2. Дотримання вимог щодо оформлення	від 0 до 10	9		Філоненко М.М.  Ніжегородцев В.О. 
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		Омельчук А.А.  Суховерхівка Л.П. 
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	47		

Оцінювання кваліфікаційної роботи науковим керівником – 47 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100- бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підписи)
94	A	5 (відмінно)	08.06.2026	Дюбанов О.С.  Філоненко М.М.  Ніжегородцев В.О.  Омельчук А.А.  Суховерхівка Л.П. 

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри  Філоненко М.М.

«01» грудня 2025

**Завдання на кваліфікаційну роботу
здобувача вищої освіти**

Павлушенка Дмитра Миколайовича

1. Тема кваліфікаційної роботи «Розробка інтерактивного веб-середовища освітнього ІТ порталу»

затверджена наказом ректора від «28» листопада 2025 №1526

2. Строк здачі закінченої роботи «08» травня 2026

3. Мета роботи: розробка веб-застосунку для інтеграції освітніх ресурсів з інтерактивними функціями відображення інформації.

Об'єкт дослідження: відкриті платформи розробки веб-середовища з інтерактивними функціями.

Предмет дослідження: Стек технологій та фреймворків React, Express.js, Node.js, PostgreSQL, функціонал середовища Replit.

Орієнтовна база дослідження: документація фреймворків React, Express.js та бібліотеки Drizzle ORM; документація платформ Oracle Academy, AWS Academy та Cisco Networking Academy; вебфоруми Stack Overflow та Reddit як платформи для аналізу практик розробки;

4. Вхідні дані до роботи: Відкриті дані та офіційна документація по React, Vite, Express.js, Node.js, Drizzle ORM, PostgreSQL, TypeScript; документація освітніх програм Oracle Academy, AWS Academy та Cisco Networking Academy;

5. Перелік графічного матеріалу: Рисунки інтерфейсів розробленого веб-застосунку; структурна схема архітектури системи (клієнт-сервер-база даних); схема бази даних (ER-діаграма таблиць користувачів, моніків прогресу)

6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1: «АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА ХМАРНИХ ПЛАТФОРМ ДЛЯ РОЗРОБКИ ОСВІТНІХ ВЕБ-РЕСУРСІВ»

1.1. Огляд та порівняльний аналіз існуючих освітніх веб-платформ

1.2. Обмеження поширених платформ для потреб університетського ІТ-хабу

1.3. Обґрунтування вибору платформи Replit та необхідності розробки власної системи

РОЗДІЛ 2: ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-СЕРЕДОВИЩА ОСВІТНЬОГО ІТ-ПОРТАЛ

2.1 Вибір технологічного стеку та архітектурних рішень

2.2 Реалізація клієнтської та серверної частин

2.3 Розгортання системи на платформі Replit

РОЗДІЛ 3. ПРОЕКТУВАННЯ СХЕМИ БАЗИ ДАНИХ ТА РЕАЛІЗАЦІЯ ДИНАМІЧНИХ (ІНТЕРАКТИВНИХ) МОЖЛИВОСТЕЙ ПЛАТФОРМИ

3.1 Проектування реляційної схеми бази даних PostgreSQL

3.2. Реалізація інтерактивного функціоналу системи

3.3. Розробка REST API та тестування системи

3.4. Огляд веб-інтерфейсів освітнього IT-порталу

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

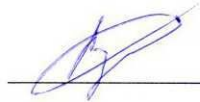
№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 30.10.2025	30.10.2025
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	26.12.2025	26.12.2025
3	II розділ (аналіз експериментальної частини про методики досліджень)	18.02.2026	18.02.2026
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	16.04.2026	16.04.2026
5	Висновки	01.05.2026	01.05.2026
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	07.05.2026	07.05.2026
7	Отримання відгуку керівника	29.05.2026	29.05.2026
8	Отримання зовнішньої рецензії	01.06.2026	01.06.2026

8. Дата видачі завдання «01» грудня 2025

9. Керівник кваліфікаційної роботи

 Редич О.В.

10. Керівник освітньо-професійної програми

 Ніжегородцев В.О.

11. Завдання до виконання отримав,
здобувач вищої освіти

 Павлушенко Д.М.

АНОТАЦІЯ

Розробка інтерактивного веб-середовища освітнього IT порталу.

Мета роботи: розробка веб-застосунку для інтеграції освітніх ресурсів трьох академічних програм (Oracle Academy, AWS Academy, Cisco Networking Academy) з інтерактивними функціями відображення інформації та підтримки навчального процесу.

Об'єкт дослідження: відкриті платформи розробки веб-середовища з інтерактивними функціями.

Предмет дослідження: процес проектування та реалізації освітнього веб-порталу з використанням стека технологій React, Express.js, Node.js та PostgreSQL.

У роботі проведено порівняльний аналіз чотирьох популярних освітніх платформ (Moodle, Google Classroom, Coursera, GitHub Education) та спеціалізованих сервісів академій-партнерів, встановлено їхні обмеження в умовах роботи багатoprogramної кафедри. На підставі аналізу обґрунтовано доцільність розробки власного веб-середовища та обрано технологічний стек: React і Vite з TypeScript на клієнті, Node.js з Express і Drizzle ORM на сервері, PostgreSQL у хмарному сервісі Neon. Спроектовано реляційну схему бази даних з шести таблиць; схема описана засобами Drizzle ORM і слугує єдиним джерелом TypeScript-типів для обох частин системи. Реалізовано клієнтську частину (14 сторінок) та серверну частину (19 REST-ендпоінтів) з модулями реєстрації й автентифікації, тестування знань, опитувань, зворотного зв'язку зі шкалою NPS, аналітичного дашборду, особистого кабінету з досягненнями та рекомендаціями й адміністративної панелі. Систему протестовано за 11 сценаріями та розгорнуто на платформі Replit.

У результаті виконання кваліфікаційної роботи розроблено й розгорнуто повнофункціональний інтерактивний освітній IT-портал кафедри KіTіC Державного податкового університету, що об'єднує три академічні програми в єдиному двомовному інтерфейсі. Портал готовий до використання в навчальному процесі кафедри.

Ключові слова: освітній веб-портал, React, Node.js, TypeScript, PostgreSQL, REST API, Drizzle ORM, LMS, NPS, інтерактивні модулі, хмарне розгортання, Replit.

Кваліфікаційна робота містить 54 сторінки, 13 рисунків, 2 таблиці, список використаних джерел 39 найменувань.

Рік виконання кваліфікаційної роботи: 2026.

Рік захисту кваліфікаційної роботи: 2026.

SUMMARY

Development of an Interactive Web Environment for an Educational IT Portal.

The aim of the work is to develop a web application for integrating the educational resources of three academic programmes (Oracle Academy, AWS Academy, Cisco Networking Academy) with interactive information display features and learning process support tools.

The object of research is open platforms for developing a web environment with interactive features.

The subject of research is the process of designing and implementing an educational web portal using a technology stack comprising React, Express.js, Node.js, and PostgreSQL.

The paper presents a comparative analysis of four widely used educational platforms (Moodle, Google Classroom, Coursera, GitHub Education) and the specialised services of partner academies, identifying their limitations in the context of a multi-programme academic department. Based on this analysis, the development of a custom web environment is justified and a technology stack is selected: React and Vite with TypeScript on the client side, Node.js with Express and Drizzle ORM on the server side, and PostgreSQL hosted in the Neon cloud service. A relational database schema consisting of six tables is designed; the schema is described using Drizzle ORM and serves as the single source of TypeScript types for both parts of the system. The client side (14 pages) and server side (19 REST endpoints) are implemented, including modules for user registration and authentication, knowledge testing (quizzes), surveys, an NPS feedback form, an analytics dashboard, a personal cabinet with achievement badges and recommendations, and an administrative panel. The system is tested against 11 functional scenarios and deployed on the Replit platform.

As a result of the qualification paper, a fully functional interactive educational IT portal for the KITiS department of the State Tax University has been developed and deployed. The portal integrates three academic programmes in

a single bilingual interface and is ready for practical use in the department's educational process.

Keywords: educational web portal, React, Node.js, TypeScript, PostgreSQL, REST API, Drizzle ORM, LMS, NPS, interactive modules, cloud deployment, Replit.

The qualification paper consists of 54 pages, 13 figures, 2 tables, list of references — 39 items.

Year of completion of the qualification paper: 2026.

Year of defending the qualification paper: 2026.

Зміст

ВСТУП.....	3
РОЗДІЛ 1. АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА ХМАРНИХ ПЛАТФОРМ ДЛЯ РОЗРОБКИ ОСВІТНІХ ВЕБ-РЕСУРСІВ.....	6
1.1. Огляд і порівняння популярних освітніх веб-платформ.....	6
1.2. Обмеження готових рішень у контексті потреб університетського ІТ-хабу	10
1.3. Обґрунтування вибору платформи Replit і розробки власної системи	12
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-СЕРЕДОВИЩА ОСВІТНЬОГО ІТ-ПОРТАЛУ	16
2.1. Вибір технологічного стеку та архітектурних рішень	16
2.2. Реалізація клієнтської та серверної частин.....	19
2.3. Розгортання системи на платформі Replit	24
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО ФУНКЦІОНАЛУ ТА API.....	26
3.1. Проєктування реляційної схеми бази даних PostgreSQL	26
3.2. Реалізація інтерактивних модулів системи	28
3.3. Розробка REST API та тестування системи.....	33
3.4. Огляд веб-інтерфейсів освітнього ІТ-порталу	36
ВИСНОВКИ.....	44
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	47

ВСТУП

Сучасна ІТ-галузь характеризується надзвичайно високими темпами оновлення технологій, що ставить перед закладами вищої освіти завдання постійно адаптовувати форми, методи та інструменти підготовки фахівців. Як зазначає В. Ю. Биков, «хмарні технології стають інфраструктурою формування відкритого освітнього середовища» [1, с. 12]. Дослідники цифровізації освіти наголошують, що ефективність підготовки ІТ-спеціалістів безпосередньо залежить від гнучкості освітнього середовища та можливості швидко інтегрувати нові курси і сертифікаційні програми [1; 2]. Одним із дієвих відповідей на цей виклик є створення спеціалізованих освітніх веб-порталів, що поєднують контент кількох програм, фіксують прогрес студентів і збирають зворотний зв'язок.

Кафедра комп'ютерних інформаційних технологій і систем факультету фінансів та цифрових технологій Державного податкового університету довгий час працює з трьома міжнародними академічними програмами — Oracle Academy, AWS Academy і Cisco Networking Academy. Кожна з них має власний навчальний контент, методiku оцінювання та систему сертифікації. До початку роботи над цим проєктом єдиної точки доступу, де здобувач міг би одночасно ознайомитись з усіма трьома напрямками, перевірити власні знання та залишити структурований відгук, не існувало

Поширені у практиці системи Moodle і Google Classroom перекривають половину поставлених завдань, проте погано адаптуються під специфіку конкретної випускової кафедри з кількома партнерськими програмами [3; 4]. Це і стало основною причиною створення власного веб-середовища.

Мета роботи — спроектувати й реалізувати інтерактивний освітній ІТ-портал, що інтегрує матеріали трьох академічних програм. М. І. Жалдак зазначав, що «комп'ютерно-орієнтоване навчальне середовище повинне органічно поєднувати традиційні форми роботи з новітніми засобами» [38, с. 7]., забезпечує облік прогресу здобувачів і аналітику якості освітнього процесу.

Завдання дослідження:

1. Провести аналіз існуючих освітніх веб-платформ, визначивши їхні недоліки та обмеження в умовах роботи багатопрограмної кафедри.
2. Надати обґрунтований вибір відповідного технологічного стеку та середовища для розгортання системи.
3. Розробити реляційну модель бази даних і сформуванати архітектуру майбутньої системи.
4. Створити клієнтський та серверний модулі, забезпечуючи повний спектр інтерактивних можливостей.
5. Провести функціональне тестування розробленої системи та підготувати її до використання у реальних умовах.

Мета роботи: розробка веб-застосунку для інтеграції освітніх ресурсів з інтерактивними функціями відображення інформації.

Об'єкт дослідження: відкриті платформи розробки веб-середовища з інтерактивними функціями .

Предмет дослідження: процес проектування та реалізації освітнього веб-порталу з використанням стека технологій React, Express.js, Node.js та PostgreSQL

Орієнтовна база дослідження: Офіційна документація фреймворків React, Express.js та бібліотеки Drizzle ORM; технічна документація платформ Oracle Academy, AWS Academy та Cisco Networking Academy; вебфоруми Stack Overflow та Reddit як майданчики для аналізу поширених практик розробки; наукові праці у галузі веб-технологій та освітніх інформаційних систем.

Вхідні дані до роботи: Відкриті дані та офіційна документація по React, Vite, Express.js, Node.js, Drizzle ORM, PostgreSQL, TypeScript; документація освітніх програм Oracle Academy, AWS Academy та Cisco Networking Academy;

Отримані результати та їх практична значимість

У результаті виконання кваліфікаційної роботи розроблено та розгорнуто повнофункціональний інтерактивний освітній IT-портал кафедри KITiC ДПУ, що об'єднує три академічні програми (Oracle Academy, AWS Academy, Cisco Networking Academy). Реалізовано модулі реєстрації та автентифікації користувачів, відстеження навчального прогресу, інтерактивного тестування, системи досягнень і

персоналізованих рекомендацій, проведення опитувань та збору структурованих відгуків з NPS-аналітикою. Практична значимість роботи полягає в тому, що портал розгорнуто на платформі Replit і готовий до використання у реальному навчальному процесі кафедри. Аналітичний дашборд з показниками NPS, радарними та лінійними графіками і хмарами ключових слів надає науково-педагогічним працівникам інструмент оперативного моніторингу якості освітнього процесу та своєчасного реагування на зворотний зв'язок здобувачів. Двомовний інтерфейс (UA/EN) забезпечує доступність платформи для різних категорій здобувачів.

Апробація. Основні результати роботи подані тезі на тему «ПРИКЛАД РОЗРОБКИ REST API ТА ТЕСТУВАННЯ СИСТЕМИ НА ПРИКЛАДІ ОСВІТНЬОГО ВЕБ-ПОРТАЛУ», опублікованій у збірнику доповідей науково-практичної конференції «Інноваційні цифрові технології: від алгоритмів до безпечних систем» Ірпінь, ДПУ, 2026 [39].

РОЗДІЛ 1. АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА ХМАРНИХ ПЛАТФОРМ ДЛЯ РОЗРОБКИ ОСВІТНІХ ВЕБ-РЕСУРСІВ

1.1. Огляд і порівняння популярних освітніх веб-платформ

Цифрова трансформація вищої освіти, активно прискорена пандемійним періодом 2020–2022 років, суттєво змінила підходи до організації навчального процесу. Ще у 2005 р. Дж. Сіменс визначав, що «здатність навчатися є важливішою, ніж те, що відомо зараз» [6, с. 5], а цифрова трансформація 2020-х надала цій тезі нового виміру. [3; 5]. Якщо ще на початку 2010-х років основним каналом передачі знань залишалась аудиторна робота, то сьогодні більша частина навчальної взаємодії здійснюється у цифровому середовищі — через системи управління навчанням, хмарні офісні екосистеми, відкриті курси та спеціалізовані освітні портали. Заклади вищої освіти України дедалі активніше впроваджують такі інструменти, що відповідає загальному курсу на цифровізацію освіти, закладеному у відповідних нормативних документах [6; 7].

Принципово важливою є зміна середовища: освітня платформа в ІТ-галузі поступово перестає бути «сховищем PDF-лекцій». Н. В. Морзе та О. Г. Кузьмінська підкреслюють, що «ефективність хмарних інструментів у навчанні визначається не функціональністю самого сервісу, а методичним сценарієм його використання» [2, с. 24]. і перетворюється на середовище взаємодії, що формує індивідуальну освітню траєкторію здобувача. Як зазначає Н. В. Морзе та інші дослідники, “якість електронного курсу нині визначається не обсягом завантажених матеріалів, а наявністю інструментів оцінювання, рефлексії та зворотного зв’язку” [2].

Ця тенденція набуває особливої уваги для ІТ-спеціальностей. Здобувачі таких напрямів уже мають достатній рівень цифрової грамотності й щодня користуються відповідними програмами. М. Пренскі ще у 2001 р. зазначав: «цифрові аборигени звикли отримувати інформацію дуже швидко» [35, с. 2] — і ця характеристика повною мірою стосується сучасних ІТ-студентів.: GitHub, Notion, Discord. Відповідно, їхні очікування до навчальної платформи включають не лише функціональність, а й сучасний візуальний дизайн, високу швидкість відгуку, інтерактивність і

персоналізацію — індикатори прогресу, рекомендації, опитування, можливість залишити відгук тощо.

Сьогоднішній ринок освітніх веб-платформ доволі насичений. Серед найпоширеніших у університетах доцільно виокремити Moodle, Google Classroom, Coursera та GitHub Education. Розглянемо їх послідовно.

Moodle (Modular Object-Oriented Dynamic Learning Environment)

Moodle — це відкрита (open-source) система управління навчанням, ініційована австралійським розробником Мартіном Дугіамасом і вперше випущена у 2002 році. За даними офіційної статистики проєкту, на 2024 рік платформою користуються понад 300 млн осіб у близько 240 країнах, що робить її найпоширенішою LMS у світі. В українських ЗВО Moodle де-факто виконує функцію базового середовища дистанційного навчання, що неодноразово фіксують вітчизняні дослідники цифрової освіти [3; 5; 7]. ДПУ також використовує Moodle для організації частини курсів.

Архітектурно система побудована на модульному принципі: адміністратор закладу може підключати лише ті компоненти, що йому потрібні — тести, форуми, модулі завдань, відеоматеріали, журнали оцінок. Платформа підтримує понад 100 мов інтерфейсу, рольову модель (здобувач, викладач, адміністратор) і ведення докладного журналу активності.

Попри поширеність, Moodle має низку суттєвих обмежень. По-перше, інтерфейс системи помітно застарів стилістично. Дж. Коул і Г. Фостер у своєму дослідженні вказують, що «Moodle дає широкі можливості, але вимагає значних зусиль адміністратора для підтримки актуального стану плагінів» [3, с. 47]. Їй поступається сучасним SPA-застосункам як за UX, так і за швидкістю роботи. По-друге, будь-яка нестандартна функція — наприклад, інтеграція з конкретною сертифікаційною програмою чи дашборд аналітики відгуків — вимагає розробки окремого PHP-плагіна, тестування на ізольованому стенді та подальшого супроводу під час оновлень ядра. По-третє, повноцінне розгортання Moodle потребує власної серверної інфраструктури або оренди VPS, що додатково навантажує IT-підрозділ ЗВО [5].

Google Classroom

Google Classroom — хмарний сервіс компанії Google, орієнтований на спрощення комунікації між викладачем і студентами. Запущений у 2014 році, він входить до екосистеми Google Workspace for Education і безкоштовно надається навчальним закладам. Через Classroom викладач створює клас, публікує матеріали, видає й приймає завдання, виставляє оцінки, надсилає сповіщення.

Сильною стороною продукту є глибока інтеграція з іншими сервісами Google — Документами, Формами, Диском, Meet, — завдяки чому курс можна сформувати буквально за лічені хвилини. Однак за функціональністю Classroom помітно поступається повноцінним LMS: тут немає системи рейтингів, бракує налаштовуваних інтерактивних тестів з аналітикою результатів, практично відсутні засоби кастомізації під специфіку кафедри. По суті, це інструмент для обміну матеріалами, а не середовище самостійного навчання з елементами гейміфікації.

Coursera

Платформа Coursera, заснована у 2012 році професорами Стенфордського університету Е. Йом та Д. Коллер, є однією з найбільших систем масових відкритих онлайн-курсів (МООС). На ній представлені програми провідних університетів і технологічних компаній, у тому числі Oracle, Amazon Web Services та Cisco. Станом на 2024 рік сервіс налічує понад 140 млн зареєстрованих користувачів.

З погляду контенту Coursera безумовно переважає інші розглянуті рішення: вона пропонує повноцінні спеціалізації та індустріальні сертифікати, які мають вагу на ринку праці [8]. Проте для університетського ІТ-хабу цей варіант підходить погано з трьох ключових причин. По-перше, переважна більшість курсів є платними. По-друге, Coursera — повністю зовнішня платформа, що не дає змоги прив'язати навчальний контент до конкретної кафедри чи академічної групи. По-третє, відсутні механізми взаємодії здобувачів у межах власного академічного контексту.

GitHub Education

GitHub Education — це освітня програма від Microsoft (власника GitHub), що надає здобувачам і викладачам ІТ-напрямів безкоштовний доступ до GitHub Pro та низки інструментів партнерських компаній. Цей сервіс не є навчальною платформою у звичному розумінні: тут немає системи курсів, тестування чи прогресу — це швидше «бандл» інструментів для технічних спеціальностей. Для студента-айтівця GitHub Education є корисним доповненням, однак не може виконувати функцію централізованого освітнього порталу кафедри.

Узагальнення основних характеристик розглянутих платформ наведено у таблиці 1.1.

Таблиця 1.1 — Порівняльний аналіз освітніх платформ за функціональністю

Критерій	Moodle	Google Classroom	Coursera	GitHub Education
Авторизація	Власна	Через Google	Власна	Через GitHub
Відстеження прогресу	Так	Частково	Так	Ні
Інтерактивні тести	Так	Частково	Так	Ні
Система рейтингів	Ні	Ні	Частково	Ні
Двомовний інтерфейс	Частково	Так	Так	Ні
Аналітика відгуків	Ні	Ні	Частково	Ні
Прив'язка до академій	Ні	Ні	Частково	Ні
Гейміфікація	Ні	Ні	Частково	Ні
Відкритий код	Так	Ні	Ні	Частково
Безкоштовно для ЗВО	Так	Так	Частково	Так
Кастомізація	Через плагіни	Мінімальна	Відсутня	Відсутня

Таблиця 1.2 — Порівняння платформ за критерієм інтерактивності

Критерій	Moodle	Google Classroom	Coursera	GitHub Education
Інтерактивні тести	Так	Частково	Так	Ні
Гейміфікація (рейтинги, бейджі)	Обмежена	Ні	Частково	Ні
Відстеження прогресу	Так	Частково	Так	Ні
Зворотний зв'язок	Форум	Ні	Відгуки	Ні

Продовження таблиці 1.2

Персоналізовані рекомендації	Ні	Ні	Частково	Ні
Двомовний контент	Частково	Так	Так	Ні
Кастомізація під кафедру	Через плагіни	Ні	Ні	Ні
Аналітика для адміністратора	Базова	Ні	Частково	Ні

Аналіз даних (табл. 1.1 та 1.2) показує, що наразі немає жодного продукту, який би повністю підійшов для роботи нашого ІТ-хабу. Найбільше нашим вимогам відповідає Coursera, але її комерційна основа та неможливість адаптації під потреби кафедри роблять цей варіант неприйнятним. Отже, вже на цьому етапі стає зрозуміло, що найкращий вихід — це розробка власної платформи.

1.2. Обмеження готових рішень у контексті потреб університетського ІТ-хабу

Якщо узагальнити висновки попереднього підрозділу й розглянути обмеження готових систем можна виокремити шість груп проблем, що роблять використання типових LMS у поточному вигляді недостатнім.

1. Відсутність прив'язки до міжнародних академічних програм

Кафедра комп'ютерних інформаційних технологій і систем виступає офіційним партнером трьох провідних міжнародних освітніх програм: Cisco Networking Academy (з 2007 р.), Oracle Academy (з 2017 р.) та AWS Academy (з 2024 р.). Кожна з них має власну сертифікаційну траєкторію, унікальний перелік дисциплін, специфічну цільову аудиторію та різний рівень складності. Жодна з розглянутих платформ не дозволяє нативно структурувати навчальний простір саме навколо цих напрямів, окремо обчислювати прогрес за кожним з них і пропонувати персоналізовані рекомендації залежно від обраної здобувачем академії.

2. Відсутність системи зворотного зв'язку та аналітики

Сучасне управління якістю освіти неможливе без систематичного збору об'єктивних даних від здобувачів [1; 8]. Стандартні LMS або взагалі не мають

інструментів зворотного зв'язку, або обмежуються найпростішими формами. Натомість для адекватної оцінки роботи освітнього хабу потрібно фіксувати щонайменше:

Загальна оцінка того, наскільки користувачі задоволені, на п'ятибальній шкалі.

2. Аналіз певних моментів, таких як якість матеріалів, рівень навчання та зручність використання платформи.

3. Визначення індексу лояльності (NPS, Net Promoter Score), який показує, наскільки користувач готовий рекомендувати програму іншим.

4. Текстові відгуки, в яких зазначені сильні сторони та поради для покращення.

5. Показ рекомендацій у динаміці з розподілом по часу та різними академіями.

6. Оновлення інформації практично в реальному часі, щоб вона залишалася актуальною.

Жодна з наявних готових платформ не може одночасно забезпечити таку глибину збору даних і високий рівень їхньої візуалізації.

3. Мовна адаптація та локалізація

Більшість МООС-платформ орієнтовані передусім на англomовну аудиторію. Хоча окремі продукти (зокрема Moodle) підтримують українську локалізацію інтерфейсу, повноцінна двомовність — з миттєвим перемиканням мови безпосередньо на сторінці, двомовним контентом академій, питаннями опитувань та підказками — вимагає окремої реалізації. Для здобувачів першого–другого курсів, у яких рівень професійної англійської ще формується, наявність україномовного інтерфейсу є критичним фактором зниження бар'єра входу.

4. Гейміфікація та мотивація

Численні дослідження освітньої психології підтверджують, що елементи гейміфікації — таблиці лідерів, система досягнень, прогрес-бари — суттєво підвищують залученість та мотивацію здобувачів. С. Детердінг та ін. визначають гейміфікацію як «використання ігрових елементів дизайну у неігровому контексті» [4, с. 10], що саме й реалізовано у розробленому порталі. [4]. Більшість стандартних LMS або зовсім не мають таких функцій, або реалізують їх у спрощеному вигляді. Для IT-спеціальностей, де ігрові механіки є частиною повсякденного цифрового досвіду, ця складова стає вагомим чинником мотивації.

5. Технічні обмеження готових рішень

Адаптація готової платформи під нові вимоги завжди нашоується на її архітектурні припущення. Реалізація нової функції в Moodle потребує написання PHP-плагіна, окремого тестового стенду та ризику конфлікту з наступними оновленнями ядра. Google Classroom і Coursera взагалі не надають доступу до вихідного коду. У власній розробці нова функція може бути додана за лічені години без ризику зачепити суміжний функціонал.

6. Розрізненість ресурсів (fragmented learning experience)

Оскільки кафедра КІТС співпрацює з кількома академіями одночасно, здобувач змушений паралельно працювати у трьох-чотирьох різних інтерфейсах: окремі портали Cisco, Oracle, AWS, плюс корпоративна Moodle університету. Це призводить до когнітивного перевантаження, потреби пам'ятати різні паролі та звикати до різних навігаційних схем. Відсутність єдиної точки входу (single entry point) безпосередньо знижує ефективність самостійної роботи. Власний портал має виконати функцію хабу, що структурує посилання, дедлайни й інструкції до всіх партнерських програм в одному місці.

1.3. Обґрунтування вибору платформи Replit і розробки власної системи

Виявлені обмеження готових рішень дають підстави стверджувати: для повноцінного задоволення потреб кафедри КІТС необхідно розробити власну веб-

платформу. Наступне питання — у виборі інфраструктури, на якій ця платформа буде створена й експлуатуватиметься.

Підходи до розгортання веб-застосунків

Класичний шлях передбачає оренду VPS, ручне налаштування ОС, веб-сервера (nginx / Apache), СКБД, SSL-сертифіката тощо. Такий процес займає від кількох годин до кількох днів і вимагає глибокої експертизи системного адміністрування. Альтернативами виступають:

1. Platform as a Service (PaaS) — Heroku, Railway, Render, що беруть на себе більшість інфраструктурних задач;
2. Cloud-based IDE — Replit, CodeSandbox, StackBlitz, які поєднують у браузері редактор коду, середовище виконання та хостинг.

Replit як середовище розробки та хостингу

Replit — хмарне середовище розробки, засноване у 2016 році. Згідно з офіційною документацією, «Replit дозволяє писати, запускати і розгортати код у браузері без жодного локального налаштування» [26]. Сьогодні платформою користується понад 30 млн розробників із 190 країн; в освітньому секторі вона активно застосовується як для викладання програмування, так і для розгортання навчальних та виробничих проєктів. Ключова концепція Replit — «налаштовано за замовчуванням»: проєкт стартує без додаткових конфігурацій сервера. Розробник лише визначає команду запуску (наприклад, `npm run dev`), а платформа автоматично забезпечує середовище виконання Node.js, мережеві налаштування і SSL-сертифікат.

Окрім інфраструктурних переваг, вибір на користь власної розробки відкриває можливість використовувати сучасний JavaScript/TypeScript-стек. На відміну від монолітних LMS попереднього покоління (Moodle), компонентний підхід React дозволяє реалізувати реактивний інтерфейс без перезавантажень сторінки. Це формує відчуття «сучасного продукту», звичне для IT-аудиторії [9].

Таблиця 1.3 — Порівняння підходів до розгортання веб-платформи

Критерій	Replit	VPS (DigitalOcean)	Railway	Heroku
Налаштування сервера	Автоматичне	Ручне	Частково	Частково
SSL-сертифікат	Вбудований	Вручну (certbot)	Вбудований	Вбудований
PostgreSQL	Вбудована (Neon)	Окремо	Окремо (платно)	Платно
Розгортання	1 клік	Скрипт	Напівавтоматичне	Автоматичне
Редактор браузері	Так	Ні	Ні	Ні
Постійний домен	Так (.replit.app)	Купується	Так	Так
Безкоштовний план	Так	Ні	Обмежено	Обмежено

Наведені оцінки показують, що використання Replit скорочує час від нуля до першого запуску застосунку з кількох годин до десяти хвилин. У навчальному проєкті це важливо: більше робочого часу залишається на саму розробку, а не на адміністрування інфраструктури.

Вбудована база даних Neon PostgreSQL

Однією з найвагоміших переваг Replit для цього проєкту є інтеграція з хмарною PostgreSQL від Neon. Згідно з документацією, Neon — це «безсерверна PostgreSQL, що автоматично масштабується до нуля під час простою» [19], що робить її ідеальним вибором для навчального проєкту. Neon — це serverless-варіант PostgreSQL, що автоматично масштабується під навантаження і не потребує ручного адміністрування. При створенні проєкту змінна DATABASE_URL генерується автоматично та одразу доступна у коді, що знімає типові кроки на кшталт окремої реєстрації у сервісі бази, налаштування мережевого доступу чи управління паролями.

Replit Deployments як механізм публікації

Для переведення проєкту у виробничий режим Replit пропонує функцію Deployments. Після публікації застосунк отримує постійний домен виду назва-проєкту--нікнейм.replit.app і працює 24/7. На відміну від режиму розробки, де процес

зупиняється при закритті браузера, розгорнутий застосунок не залежить від активності робочого простору.

Підсумовуючи розділ, зауважимо: жодна з розглянутих готових платформ не закриває одночасно всі вимоги до централізованого ІТ-хабу прив'язку до трьох академій, двомовний інтерфейс, інтерактивний облік прогресу, систему опитувань і аналітику відгуків, гейміфікацію. Розробка власного рішення на сучасному JavaScript-стеку з розгортанням у хмарному середовищі Replit дозволяє не лише закрити всі ці потреби, а й забезпечити запас гнучкості для майбутнього розширення системи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-СЕРЕДОВИЩА ОСВІТНЬОГО ІТ-ПОРТАЛУ

2.1. Вибір технологічного стеку та архітектурних рішень

Перш ніж розпочати кодування, було необхідно зафіксувати, на якому стеку технологій будуватиметься проєкт. Питання неоднозначне: варіантів багато, і кожен має сильні та слабкі сторони залежно від характеру завдань. Основна вимога до системи — інтерактивність: портал має не просто демонструвати статичні описи академій, а дозволяти реєстрацію, проходження тестів, відстеження прогресу, голосування в опитуваннях і залишання структурованих відгуків. Це одразу визначило формат — повноцінний веб-застосунок з клієнтською та серверною частинами і реляційною базою даних.

Мова програмування

Для обох сторін — і фронтенду, і бекенду — обрано TypeScript. Згідно з даними Stack Overflow Developer Survey 2024, TypeScript посідає п'яте місце серед найпопулярніших мов програмування і третє — серед тих, які розробники хотіли б використовувати [39]. TypeScript розширює JavaScript статичною типізацією. Офіційна документація визначає TypeScript як «JavaScript із синтаксисом для типів» [27], що дозволяє виявляти помилки ще на етапі компіляції, до запуску застосунку.: він дозволяє оголошувати типи змінних і перевіряти коректність коду ще до запуску. Ключова перевага такого вибору — можливість описати схему даних один раз у каталозі `shared/` і використовувати її одночасно на сервері та клієнті. Опис таблиці `users` у файлі `shared/schema.ts` автоматично генерує типи і для форми реєстрації на фронтенді, і для функцій роботи з базою на бекенді. Це усуває цілий клас помилок, коли клієнт очікує одну структуру, а сервер повертає іншу.

Клієнтська частина

Для фронтенду застосовано бібліотеку React у поєднанні з інструментом збірки Vite. React ґрунтується на компонентному підході: інтерфейс розбивається на самостійні фрагменти. Документація React визначає компонент як «незалежний та

перевикористовуваний фрагмент UI» [13], що значно полегшує розробку і тестування складних інтерфейсів. (картка академії, форма відгуку, таблиця лідерів), кожен з яких можна розробляти й тестувати незалежно. Vite забезпечує миттєве оновлення інтерфейсу у браузері без перезавантаження сторінки завдяки технології Hot Module Replacement (HMR).

Маршрутизацію між сторінками реалізовано через Wouter — легкий аналог React Router (≈ 2 КБ), якого цілком достатньо для проєкту такого масштабу. Для отримання даних із сервера й управління кешуванням обрано TanStack Query. Документація визначає його як інструмент для «синхронізації, кешування та оновлення серверного стану у веб-застосунках» [20].: він автоматично кешує відповіді, відстежує стани завантаження та повторно запитує дані з заданим інтервалом. Наприклад, дашборд аналітики відгуків самостійно оновлюється кожні 30 секунд без ручного коду опитування.

Стилізація побудована на Tailwind CSS у поєднанні з бібліотекою компонентів shadcn/ui — стилі задаються одразу в розмітці через утилітарні класи, а готові компоненти забезпечують консистентність UI. Анімації реалізовано через Framer Motion: плавні переходи й появи елементів описуються кількома рядками декларативного коду.

Серверна частина

Бекенд реалізований на Node.js з фреймворком Express.js. Node.js дозволяє зберігати єдину мову програмування на обох рівнях; Express.js — мінімалістичний інструмент для побудови HTTP-серверів і REST API. Для роботи з PostgreSQL використано Drizzle ORM — типобезпечний інструмент, у якому замість «сирого» SQL-запиту пишеться типізований TypeScript-код, що перевіряється компілятором. Автентифікацію забезпечує Passport.js з локальною стратегією, сесії зберігаються у PostgreSQL через connect-pg-simple. Валідація вхідних даних здійснюється через Zod, схеми якого узгоджуються з drizzle-zod.

Узагальнений технологічний стек подано в таблиці 2.1.

Таблиця 2.1 — Технологічний стек освітнього IT-порталу ДПУ

Шар системи	Технологія	Версія	Призначення
Мова програмування	TypeScript	5.x	Єдина мова для фронтенду й бекенду, статична типізація
Фронтенд	React	18.x	Побудова компонентного інтерфейсу
Збірка	Vite	5.x	Швидкий dev-сервер і збірка статичних файлів
Маршрутизація	Wouter	3.x	Клієнтська навігація між сторінками
Кешування запитів	TanStack Query	5.x	Стан сервера, кеш, автооновлення
Стилізація	Tailwind CSS	3.x	Утилітарні класи, адаптивний дизайн
UI-компоненти	shadcn/ui	—	Готові доступні компоненти
Анімації	Framer Motion	11.x	Плавні переходи та появи
Бекенд	Express.js	4.x	HTTP-сервер і REST API
Runtime	Node.js	20.x	Середовище виконання JS на сервері
ORM	Drizzle ORM	0.30.x	Типобезпечна робота з PostgreSQL
База даних	PostgreSQL (Neon)	16.x	Реляційне хмарне сховище даних
Автентифікація	Passport.js	0.7.x	Локальна стратегія login/password
Сесії	connect-pg-simple	—	Збереження сесій у PostgreSQL
Валідація	Zod + drizzle-zod	—	Схеми валідації вхідних даних
Форми	react-hook-form	7.x	Керовані форми з валідацією

Архітектура системи

Систему побудовано за монолітною клієнт-серверною архітектурою: фронтенд і бекенд розгортаються як єдиний застосунок, що працює в одному процесі. Для проєкту масштабу освітнього порталу кафедри мікросервісний підхід є надмірно складним і не виправданим. Завдяки спільному каталогу `shared/` типи TypeScript не дублюються — описані один раз, вони використовуються одночасно на сервері та клієнті.

Продовження таблиці 2.2

/leaderboard	Таблиця лідерів
/feedback	Форма зворотного зв'язку
/analytics	Дашборд аналітики відгуків
/news	Новини
/profile	Особистий кабінет із досягненнями
/admin	Панель адміністратора
/login	Сторінка входу
/register	Сторінка реєстрації

Найбільшою за обсягом є головна сторінка: вона складається з дванадцяти секцій — <герой-банера> з плаваючими чіпами академій, карток цільових аудиторій, анімованих лічильників статистики, картки прогресу авторизованого користувача, карток трьох академій, блоку переваг, покрокової інструкції, блоку спільноти з активним опитуванням і топ-5 таблиці лідерів, календаря подій на 2026 рік, стрічки партнерів та фінального заклику до дії.

Усі анімовані елементи — появи блоків при прокручуванні сторінки, плаваючі чіпи академій, лічильники — реалізовані засобами “Framer Motion” . Числові лічильники використовують хук “useInView” для відстеження моменту появи елемента у полі зору та “requestAnimationFrame” для плавної анімації від 0 до цільового значення. На головній сторінці одночасно відображається чотири таких лічильники: кількість зареєстрованих користувачів, кількість тем, кількість пройдених модулів і відсоток безкоштовного доступу.

Двомовність реалізовано через контекст “LanguageContext”. При натисканні перемикача UA/EN змінюється мова, значення у контексті, і всі компоненти автоматично перемальовуються з текстом відповідною мовою. Самі переклади зберігаються безпосередньо у компонентах у вигляді умовного виразу `lang === "uk"` .

На практиці реалізація двомовності виявилась простішою, ніж здавалось на початку. Я спробував кілька підходів — окремі файли перекладів у форматі JSON, бібліотеку `react-i18next` — але зрештою обрав найпростіший варіант: умовний вираз безпосередньо у JSX. Для проєкту такого масштабу, де контент стабільний і не

надходить з CMS, це цілком виправдано: менше залежностей, нема ризику втрати ключа перекладу під час компіляції.

Контекст мови є глобальним — він оголошений у файлі `client/src/contexts/LanguageContext.tsx` і обгорнутий навколо всього дерева компонентів у `App.tsx`. Це означає, що перемикання відбувається миттєво: React перемальовує тільки ті компоненти, які підписані на контекст, не торкаючись решти. На практиці затримка між натисканням кнопки UA/EN і зміною тексту на всій сторінці непомітна — вона менша за кадр анімації.

Щодо адаптивності: усі сторінки проектувались одразу з урахуванням мобільних пристроїв, а не додавались пізніше. Tailwind CSS із його `system of responsive prefixes` (`sm:`, `md:`, `lg:`) дозволяє описувати поведінку на різних розмірах екрана прямо в `className`. Наприклад, трьохколонкова сітка карток академій на великих екранах записується як `«grid grid-cols-1 md:grid-cols-3»`, і ніяких медіа-запитів у CSS-файлах вже не потрібно. Під час тестування на реальних смартфонах жоден із основних блоків не виїжджав за межі екрана і не вимагав горизонтального скролу.

Адаптивний дизайн забезпечують утилітарні класи Tailwind CSS із префіксами `md:` і `lg:`. На мобільних пристроях багатоколонкові сітки автоматично переходять в одноколонкові, а суто декоративні елементи приховуються для зменшення візуального шуму.

Усі форми побудовано за допомогою “`react-hook-form`” у зв’язці з “`zodResolver`”: схема валідації описується один раз через `Zod` і застосовується до форми автоматично. У разі помилки користувач одразу бачить повідомлення поруч із полем. Найскладнішою у системі є форма відгуку (`/feedback`): вона включає вибір академії, загальну оцінку від 1 до 5 зірок, три окремі оцінки за критеріями, NPS-шкалу 0–10, перемикач загального враження та умовні текстові поля, що з’являються залежно від обраного враження.

Серверна частина: API та бізнес-логіка

Бекенд реалізовано як REST API. Ключове архітектурне рішення — введення абстракції `IStorage`. Це TypeScript-інтерфейс, що описує всі операції з даними; маршрути у `routes.ts` звертаються не до бази напряду, а до методів об'єкта `storage`. Завдяки такому розподілу маршрути залишаються «тонкими», а вся логіка роботи з даними зосереджена в одному місці і може бути замінена іншою реалізацією без переписування ендпоінтів.

Таблиця 2.3 — Перелік API-ендпоінтів системи

Метод	Маршрут	Призначення	Доступ
POST	/api/register	Реєстрація нового користувача	Публічний
POST	/api/login	Вхід у систему	Публічний
GET	/api/logout	Вихід із системи	Авторизований
GET	/api/user	Дані поточного користувача	Авторизований
PATCH	/api/profile	Оновлення профілю	Авторизований
GET	/api/topics	Список усіх академій	Публічний
GET	/api/topics/:slug	Деталі конкретної академії	Публічний
GET	/api/progress	Прогрес поточного користувача	Авторизований
POST	/api/quiz/result	Збереження результату тесту	Авторизований
GET	/api/users/leaderboard	Таблиця лідерів	Публічний
GET	/api/polls	Активне опитування	Публічний
POST	/api/polls/:id/respond	Голосування в опитуванні	Публічний
POST	/api/feedback	Надсилання відгуку	Публічний
GET	/api/feedback	Список усіх відгуків	Публічний
GET	/api/feedback/analytics	Аналітика відгуків з фільтрами	Публічний
GET	/api/users/me/achievements	Список досягнень користувача	Авторизований
GET	/api/users/me/recommendations	Персональні рекомендації	Авторизований
GET	/api/admin/feedback.csv	Експорт відгуків у CSV	Адміністратор
DELETE	/api/admin/feedback	Очищення усіх відгуків	Адміністратор

Автентифікація реалізована через `Passport.js` з локальною стратегією. Сесії зберігаються у таблиці `sessions` у PostgreSQL — при перезапуску сервера авторизовані

користувачі не втрачають доступ, оскільки їхні сесії відновлюються з бази втоматично. Термін дії сесії становить 30 діб. Захист авторизованих маршрутів забезпечує middleware-перевірка `req.isAuthenticated()`; для адміністративних ендпоінтів додатково контролюється поле `isAdmin` у профілі користувача.

Найскладнішим за бізнес-логікою є ендпоінт `"/api/feedback/analytics"`. Він підтримує фільтри за часовим періодом (7 днів / 30 днів / увесь час) та академією і повертає:

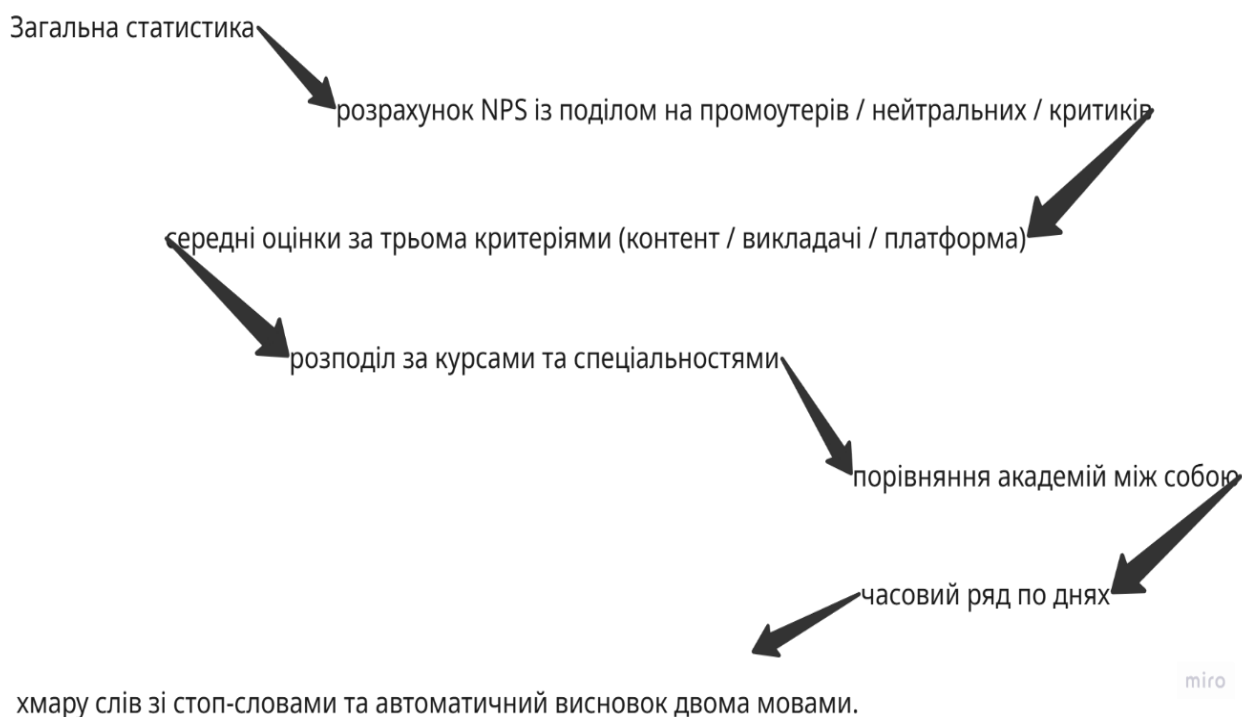


Рисунок 2.2 — Схема ендпоінту `/api/feedback/analytics`

Особистий кабінет користувача надає доступ до двох додаткових функцій. Система досягнень побудована без окремої таблиці в базі — 10 бейджів обчислюються динамічно на сервері за наявними даними: за факт реєстрації, за завершення першого тесту, за проходження всіх трьох академій, за залишений відгук тощо. Кожен бейдж має назву й опис двома мовами. Система рекомендацій аналізує прогрес і повертає до трьох не пройдених тем із поясненням, чому саме їх варто пройти наступними.

2.3. Розгортання системи на платформі Replit

Запуск застосунку у Replit здійснюється через механізм Workflows — попередньо налаштованих команд. Для цього проєкту визначено один workflow «Start application» з командою “npm run dev”, яка одночасно запускає Express-сервер і Vite-сервер. Обидва процеси слухають один порт (5000): Vite підключається як middleware всередині Express, що знімає потребу в проксіюванні.

Усі чутливі параметри рядок підключення до бази, секрет для шифрування сесій зберігаються у змінних середовища Replit і не потрапляють у вихідний код. Підключення до Neon PostgreSQL виконується через пул з’єднань:

```
import { Pool } from "pg";

const pool = new Pool({ connectionString: process.env.DATABASE_URL });
```

Пул з’єднань — це механізм повторного використання вже відкритих підключень: сервер забирає з пулу вільне з’єднання, виконує запит і повертає його, що знижує затримки та навантаження на базу.

Для синхронізації схеми таблиць із описом у shared/schema.ts використовується команда npm run db:push. Вона аналізує описану через Drizzle ORM схему та застосовує до бази необхідні зміни — додає нові стовпці, змінює типи, створює відсутні таблиці — без необхідності писати SQL-міграції вручну.

Щоб платформа виглядала наповненою з першого запуску, реалізовано механізм idempotent-seed: при кожному старті сервера виконується перевірка наявності трьох академій, 14 демо-акаунтів та демо-відгуків. Записи, яких бракує, доповнюються; ті, що вже існують, не дублюються. Завдяки цьому після розгортання на новому середовищі портал одразу демонструє реалістичну картину використання.

Replit Deployments — функція публікації застосунку на постійний домен з безперервною роботою (24/7). На відміну від режиму розробки, де сервер зупиняється при закритті браузера, розгорнутий застосунок працює автономно.

Таблиця 2.4 — Порівняння режимів роботи на Replit

Характеристика	Режим розробки	Replit Deployments
Доступність	При відкритому браузері	24/7
URL	Тимчасовий	Постійний (.replit.app)
База даних	Dev-середовище	Окреме production-середовище
Змінні середовища	Dev-конфігурація	Production-конфігурація
Оновлення коду	Автоматично при збереженні	Після ручної публікації
Простий при оновленні	—	Нульовий (blue-green deploy)
Призначення	Розробка і тестування	Кінцеві користувачі

Після публікації освітній IT-портал ДПУ став доступний за адресою `dru-hub-2--dimaravlyshenko.replit.app`. При наступних оновленнях розгортання відбувається без простою: нова версія готується паралельно зі старою і підмінює її лише після успішного запуску (blue-green deployment). Replit Deployments підтримує роздільні змінні середовища для розробки і продакшну, тож тестові дані та демо-записи не змішуються з реальними.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО ФУНКЦІОНАЛУ ТА API

3.1. Проєктування реляційної схеми бази даних PostgreSQL

Перш ніж переходити до програмування інтерактивних функцій, необхідно чітко визначити, які дані зберігає система та якими зв'язками вони пов'язані. Дж. Бігз і К. Танг наголошують, що «добре спроектоване навчальне середовище відповідає принципу конструктивного вирівнювання між цілями, діяльністю та оцінюванням» [32, с. 97]. та якими зв'язками вони пов'язані. Це етап, на якому недбалість може погано закінчитись : погано спроектована схема означає, що при додаванні нових функцій доведеться переробляти половину бази та все , що на ній тримається.

Системою керування базою даних обрано PostgreSQL, розгорнута у хмарному сервісі Neon. PostgreSQL — одна з найбільш просунутих реляційних СУБД з відкритим кодом; документація визначає її як систему, що «надає надійне, продуктивне зберігання даних» [18]. Реляційна модель тут підходить краще за NoSQL-альтернативи: дані добре структуруються, є прямі зв'язки між сутностями, а PostgreSQL підтримує повний набір агрегатних функцій, необхідних для аналітики відгуків [10; 18]: дані добре структуруються, є прямі зв'язки (користувач проходить академію, користувач відповідає на опитування), а PostgreSQL чудово справляється з агрегатними запитами, необхідними для аналітики відгуків [10].

Схема описана у файлі “shared/schema.ts” через Drizzle ORM — прямо у TypeScript-кодi. Такий підхід дає автоматичну типізацію та усуває дублювання: одне й те саме оголошення таблиці одночасно є і схемою бази даних, і TypeScript-типом для фронтенду й бекенду.

Усього у базі сформовано шість основних таблиць. Центральною є users, навколо якої будується весь інтерактив системи. Її структура показана у таблиці 3.1.

Таблиця 3.1 — Структура таблиці user

Поле	Тип	Обов'язкове	Опис
Id	SERIAL	Так	Унікальний ідентифікатор
Username	VARCHAR	Так	Логін
Password	VARCHAR	Так	Хеш пароля
Display Name	VARCHAR	Ні	Ім'я для відображення
Email	VARCHAR	Ні	Електронна пошта
Course	VARCHAR	Ні	Курс навчання
Specialty	VARCHAR	Ні	Спеціальність
IsAdmin	BOOLEAN	Так	Ознака адміністратора
CreatedAt	TIMESTAMP	Так	Дата реєстрації

Таблиця “topics” зберігає інформацію про три академії. Поле “subTopics” реалізоване як JSONB і містить масив підтем із двомовними назвами, описами та посиланнями на матеріали

Таблиця progress фіксує, яку академію та з яким результатом пройшов кожен користувач (таблиця 3.2).

Таблиця 3.2 — Структура таблиці progress

Поле	Тип	Обов'язкове	Опис
id	SERIAL	Так	Унікальний ідентифікатор
userId	INTEGER	Так	Зовнішній ключ → users.id
topicId	INTEGER	Так	Зовнішній ключ → topics.id
completed	BOOLEAN	Так	Ознака завершення
score	INTEGER	Ні	Результат тесту у відсотках
completedAt	TIMESTAMP	Ні	Дата завершення

Найбільшою за кількістю полів є таблиця “feedback_responses”, що зберігає структуровані відгуки — від загальної оцінки до окремих критеріїв і текстових

коментарів. Поле “userId” залишене необов’язковим: система приймає й анонімні відгуки без авторизації.

Таблиця 3.3 — Структура таблиці feedback_responses

Поле	Тип	Опис
id	SERIAL	Унікальний ідентифікатор
userId	INTEGER	Зовнішній ключ → users.id (nullable)
name / course / specialty	VARCHAR	Профіль автора
topic	VARCHAR	Академія: oracle amazon / cisco / general
ratingOverall	INTEGER	Загальна оцінка 1–5
ratingContent ratingTeachers ratingPlatform	INTEGER	Оцінки за трьома критеріями
npsScore	INTEGER	NPS-оцінка 0–10
liked	BOOLEAN	Загальне враження
positiveAspects negativeAspects feedbackText	TEXT	Текстові коментарі
createdAt	TIMESTAMP	Дата надсилання

Для блоку спільноти на головній сторінці передбачено дві пов’язані таблиці — “polls” і “poll_responses”. Перша зберігає питання й варіанти відповідей у JSONB-полі, друга — вибір конкретного користувача. Завдяки такому поділу поточний розподіл голосів видається без перерахунку «з нуля» на кожному запиті.

Таблиця “sessions” має технічне призначення — її автоматично обслуговує бібліотека “connect-pg-simple”. У ній зберігаються активні сесії: ідентифікатор, серіалізовані дані користувача і термін дії. Завдяки цьому перезапуск сервера не примушує користувачів повторно входити: їхні сесії залишаються у базі та відновлюються автоматично.

Усі таблиці, де присутнє поле “userId”, оголошено з опцією onDelete: “cascade” — при видаленні користувача його прогрес і голоси видаляються разом із обліковим записом. Відгуки навпаки, зберігаються: вони мають аналітичну цінність навіть без прив’язки до конкретного акаунта.

3.2. Реалізація інтерактивних модулів системи

Інтерактивність — це те, що виділяє та виокремлює сайт для легкого та ефективного користування. Кожна функція, у якій бере участь користувач, реалізована як замкнений цикл:

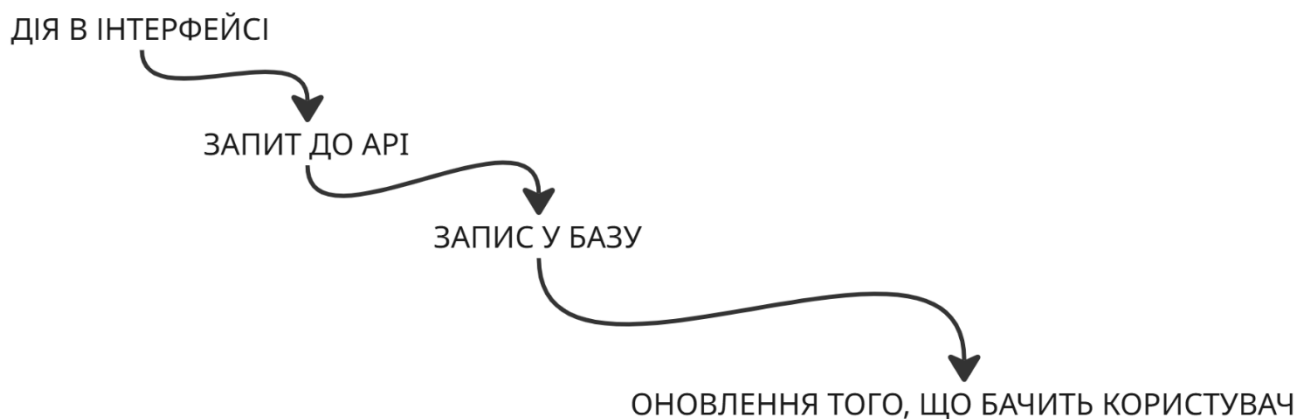


Рисунок 3.1 — Схема тестування знань

Тести доступні для кожної з трьох академій за маршрутом “/quiz/:slug”.. Питання відображаються по одному; після відповіді йде наступне. Після останнього питання обчислюється відсоток правильних відповідей і надсилається на сервер через POST “/api/quiz/result”. Якщо тест уже проходився раніше, результат просто оновлюється (дублікатів у базі не виникає). На сторінці академії стан проходження відображається одразу: завершена позначається зеленим бейджем з результатом, незавершена закликом пройти тест. Ці дані підвантажуються через GET “/api/progress” і кешуються TanStack Query на п’ять хвилин, тож зайвих запитів при навігації не виникає.

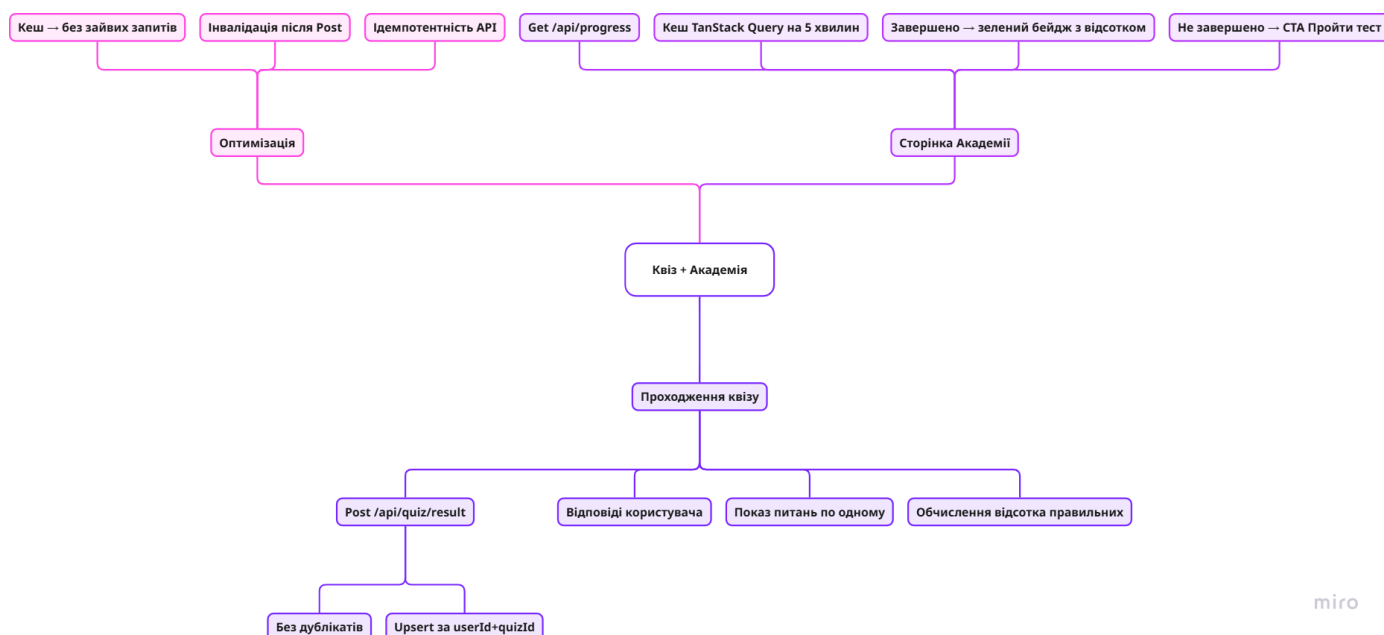


Рисунок 3.2 — Структура таблиці лідерів

На сторінці “/leaderboard” відображається рейтинг за кількістю пройдених академій. Сервер об’єднує таблиці `users` і `progress`, обчислює завершені теми кожного користувача і повертає відсортований список. Перші три позиції візуально підсвічуються кольорами медалей. Таблиця оновлюється при кожному переході на сторінку та після успішного проходження тесту (через інвалідацію відповідного кешу).

Опитування спільноти

На головній сторінці активне опитування завантажується через GET “/api/polls”. При виборі варіанту надсилається запит POST “/api/polls/:id/respond”, і одразу результати оновлюються у вигляді прогрес-барів. Повторне голосування блокується на рівні сервера: якщо для цього користувача й цього опитування вже існує запис, сервер повертає відмову.

Додатково розглянемо механізм захисту від накручування результатів. Сервер перевіряє пару (`userId`, `pollId`) у таблиці `poll_responses` перед збереженням нового голосу. Якщо запис уже існує, повертається HTTP 409 Conflict із зрозумілим повідомленням. Клієнт у відповідь показує toast-сповіщення і не оновлює локальний стан. Такий підхід є кращим, ніж блокування на рівні UI, — JavaScript у браузері можна обійти, тоді як серверна перевірка залишається надійною.

Особливого підходу вимагала NPS-шкала у формі відгуку. Стандартні бібліотеки компонентів не мають готового NPS-поля, тому його довелось реалізувати власноруч. Шкала складається з 11 кнопок (від 0 до 10), розфарбованих у три зони: 0–6 — відтінки червоного (детрактори), 7–8 — жовтий (пасиви), 9–10 — відтінки зеленого (промоутери). При наведенні кнопки злегка збільшуються завдяки Tailwind-класу `hover:scale-105`. Після вибору кнопка залишається натиснутою і підсвічується яскравіше — так користувач завжди бачить, яке значення вже обрав.

Логіка обчислення Net Promoter Score на сервері стандартна: $NPS = \text{частка промоутерів (9–10)} - \text{частка детракторів (0–6)}$, помножена на 100. Результат варіюється від -100 до +100. Для навчальної платформи значення вище +50 прийнято вважати відмінним. Поточні демо-дані, засіяні при першому запуску, дають NPS близько +43, що відповідає рівню «добре» і виглядає реалістично для платформи на ранньому етапі.

Форма зворотного зв'язку

Форма `/feedback` містить зірковий рейтинг за чотирма критеріями, NPS-шкалу з 11 кнопок з кольоровим зонуванням та умовні текстові поля, які з'являються чи зникають залежно від обраного загального враження. Усі поля валідуються локально через Zod ще до надсилання — у разі помилки користувач одразу бачить підказку, не чекаючи відповіді сервера. Після успішного збереження з'являється toast-сповіщення з підтвердженням.

Дашборд аналітики

Дашборд `/analytics` — найнасиченіший за обсягом відображуваних даних елемент системи. Він отримує дані з `GET /api/feedback/analytics` і показує: п'ять KPI-карток, кругову діаграму NPS, радарну діаграму оцінок за критеріями, лінійний графік часового ряду, стовпчасту діаграму порівняння академій, дві хмари слів і сортовану таблицю відгуків. Дані оновлюються автоматично кожні 30 секунд через параметр `refetchInterval` у TanStack Query. При зміні фільтрів за діапазоном дат або академією запит надсилається повторно з новими параметрами.

Окрема функція — експорт у PDF. Бібліотеки `html/2/canvas` і `js.pdf` не вантажаться заздалегідь, а імпортуються динамічно лише в момент натискання кнопки. Весь дашборд рендериться у `canvas` і зберігається як PDF-файл. Такий підхід не збільшує час першого завантаження сторінки.



Рисунок 3.3 — Система досягнень і рекомендацій

В особистому кабінеті `/profile` є два блоки, що прямо стимулюють подальшу взаємодію з платформою. Досягнення — це 10 бейджів з різними умовами отримання, що обчислюються динамічно на сервері без окремої таблиці у базі.

Таблиця 3.4 — Перелік досягнень системи

Бейдж	Умова отримання
Перший крок	Реєстрація в системі
Перший тест	Завершення будь-якого тесту
Oracle Scholar	Проходження Oracle Academy
AWS Pioneer	Проходження AWS Academy
Cisco Expert	Проходження Cisco Networking Academy
Тріада знань	Проходження всіх трьох академій
Голос спільноти	Участь в опитуванні
Зворотний зв'язок	Надсилання відгуку
Лідер	Входження до топ-5 таблиці лідерів

Рекомендації — до трьох не пройдених тем із поясненням, чому саме їх варто пройти наступними. Логіка враховує вже завершені академії і пропонує послідовне продовження. Обидва блоки повертаються двома мовами.

3.3. Розробка REST API та тестування системи

Структура API

Серверна частина побудована як REST API на Express.js. Основним рішенням було—розподіл між обробкою HTTP-запитів і роботою з базою даних. Для цього у `server/storage.ts` введено інтерфейс `IStorage`: він описує всі операції з даними, а клас `DatabaseStorage` реалізує їх через Drizzle ORM. Маршрути в `routes.ts` не містять SQL-логіки — вони отримують запит, валідуються Zod-схемою та викликають потрібний метод `storage`. Такий поділ відповідальностей дозволяє змінювати кожну частину незалежно.

Найскладнішим є `GET /api/feedback/analytics`: він не просто вибирає записи, а обчислює NPS, будує хмару слів зі стоп-словами для двох мов і формує автоматичний висновок, тональність якого визначається значенням NPS. Адмін-ендпоінт експорту CSV додає UTF-8 BOM на початок файлу — без цього Microsoft Excel некоректно відображає кириличні символи.

Тестування

Тестування проводилось вручну паралельно з розробкою: після реалізації кожного нового блоку він одразу перевірявся за основними сценаріями використання. Такий підхід дозволяв виявляти помилки до того, як вони могли зачепити суміжний функціонал.

Перевірка граничних випадків проводилась окремо від основних сценаріїв. Зокрема, тестувалось подання форми відгуку з NPS поза діапазоном 0–10: Zod-схема на сервері відхиляє такі дані на рівні валідації, до запиту до бази справа не доходить, і клієнт отримує HTTP 400 з масивом помилок валідації. Аналогічно перевірялось пряме звернення до захищених маршрутів без авторизації — у всіх випадках сервер повертав 401 Unauthorized із JSON-відповіддю, а не порожню сторінку або HTML-виняток.

Ідемпотентність механізму ініціалізації перевірялась неодноразово: сервер перезапускався 5–6 разів поспіль, і кожного разу кількість записів у таблицях `topics`, `users` та `feedback_responses` залишалась незмінною. Демо-дані дозаповнювались лише тоді, коли їх бракувало — наприклад, після ручного видалення одного з тестових відгуків. Це критично для середовища, де розробник може необачно очистити базу і хоче одразу повернутись до робочого стану.

Перевірка адаптивності проводилась через вбудований інструмент DevTools у Chrome: сторінки тестувались при ширині 375 px (iPhone SE), 768 px (iPad) та 1280 px (ноутбук). Жоден із блоків не виходив за межі вьюпорту, скр^У залишався лише вертикальним. Навігаційне меню на мобільних пристроях автоматично переходить у компактний вигляд завдяки класам `hidden md:flex`.

Продуктивність завантаження перевірялась через вкладку Lighthouse у Chrome DevTools. Показник Performance на головній сторінці становив 87/100 у режимі desktop. Основний час завантаження займає бандл React, однак завдяки Vite і code-splitting компоненти завантажуються по мірі потреби, а не всі одразу. Найтяжчий модуль — аналітичний дашборд із Recharts — ізольований на окремому маршруті й не гальмує початкове завантаження порталу.

Таблиця 3.5 — Результати функціонального тестування

Сценарій	Очікуваний результат	Статус
Реєстрація нового користувача	Запис у БД, автоматичний вхід	Пройдено
Вхід з правильними даними	Активна сесія, перенаправлення	Пройдено
Вхід з неправильним паролем	Повідомлення про помилку	Пройдено
Пройдення тесту	Збереження результату, оновлення прогресу	Пройдено
Надсилання відгуку	Запис у БД, підтвердження	Пройдено
Голосування в опитуванні	Оновлення результатів без перезавантаження	Пройдено
Повторне голосування	Відмова сервера	Пройдено

Продовження таблиці 3.5

Аналітика з фільтром	Коректні дані для обраного діапазону	Пройдено
Доступ до адмін-панелі без прав	Відмова з кодом 403	Пройдено
Перемикання мови	Миттєва зміна всіх текстів	Пройдено
Адаптивність на мобільному	Коректне відображення без скролу	Пройдено

Програма була випробувана у випадках тех-краплення, як заповнювання форм відсутньою інформацією, NPS поза межі 0-10 діапазону, і прохання про захищені шляхи без входу в кожну ситуацію, сервер правильно відмовився надавати належний стан HTTP і чисте повідомлення без дотику бази даних Ідіпотент було перевірено самостійно: навіть після перезапуску декількох серверів, кількість записів у таблицях тем, а користувачі залишалися однаковими. і демо-дані були додані тільки тоді, коли їх не було поряд

Перспективи розвитку

Попри загальну готовність системи до реального використання, є кілька напрямів, що можуть суттєво посилити платформу надалі.

По-перше, варто запровадити автоматизовані тести. Зараз усі перевірки проводяться вручну, що займає час і не гарантує відсутності регресій після значних змін. Для REST API підійдуть інтеграційні тести на базі supertest — вони дозволяють перевіряти кожен ендпоінт ізольовано, підмінюючи базу тестовою схемою. Для клієнтських компонентів доречно використати Vitest у парі з React Testing Library.

По-друге, перспективним є впровадження сповіщень електронною поштою: підтвердження реєстрації, нагадування про незавершені академії, щомісячний дайджест для викладачів з аналітикою відгуків за підрозділ. Такі функції можна реалізувати через Resend або SendGrid — обидва сервіси мають безкоштовний план, достатній для навчальної аудиторії.

По-третє, інтеграція OAuth-авторизації через університетський акаунт Google (Google Workspace for Education) знизить бар'єр входу для нових користувачів: замість окремої

реєстрації студент зможе увійти в один клік, і його email і ім'я заповняться автоматично. Технічно це вимагає додавання passport-google-oauth20 та реєстрації OAuth-застосунку в Google Cloud Console.

3.4. Огляд веб-інтерфейсів освітнього IT-порталу

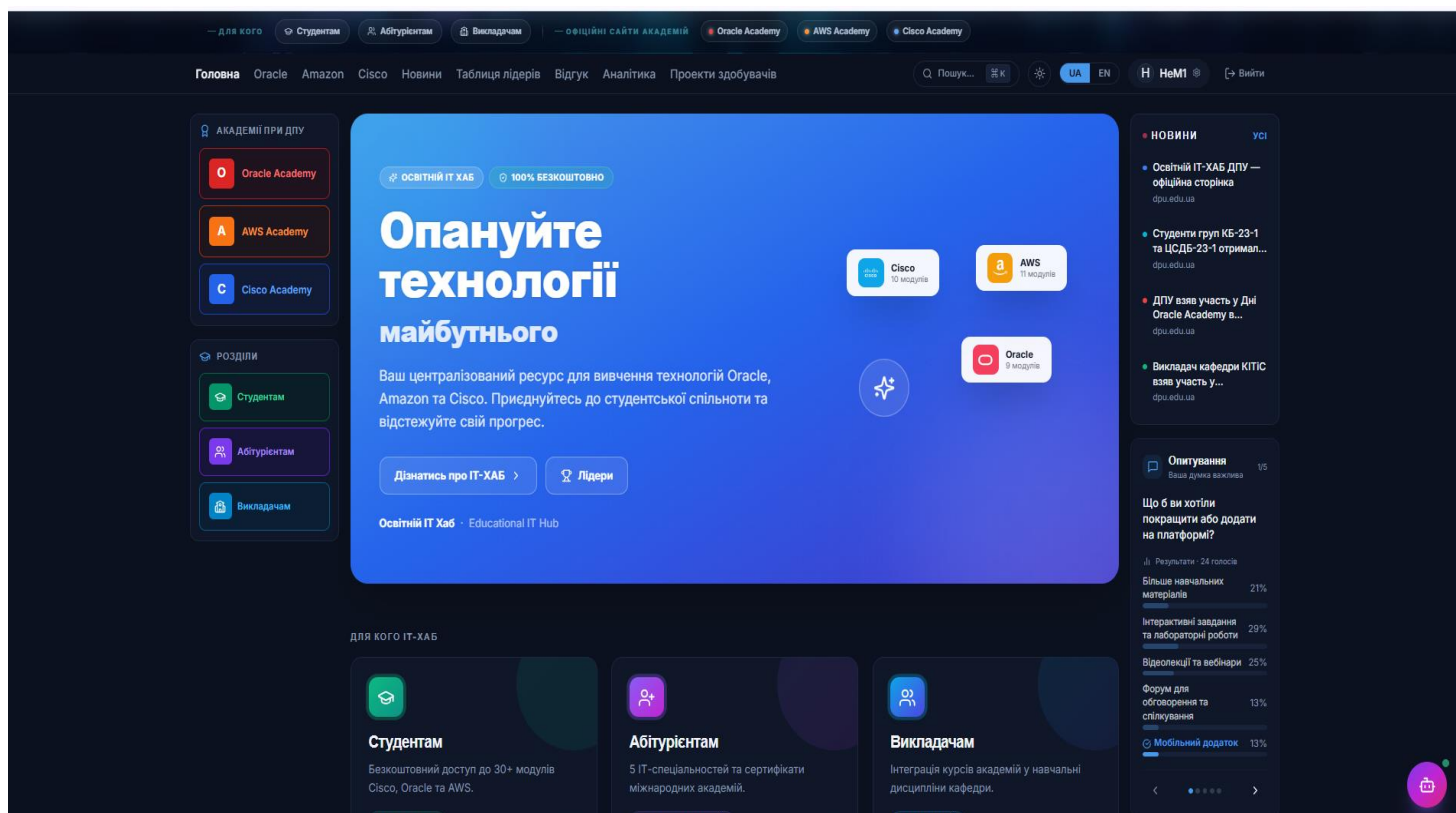


Рисунок 3.3 — (Веб-інтерфейс сайту)

Зручний та інтуїтивний веб-інтерфейс є одним із ключових чинників успішності освітньої платформи. Р. Майєр доводить, що «люди вчаться глибше зі слів та малюнків, ніж лише зі слів» [34, с. 3], тому кожна сторінка порталу поєднує текстовий контент з інтерактивними візуальними елементами. У процесі розробки особлива увага приділялась адаптивності, двомовності та візуальній ієрархії сторінок. Нижче описано основні сторінки порталу, їхній зовнішній вигляд та функціональне призначення.

При розробці кожної сторінки застосовувалась єдина концепція: максимум корисної інформації при мінімальній когнітивній навантаженості. Це передбачало чітку візуальну ієрархію (великий заголовок → підзаголовок → текст → кнопки дій), однорідну систему відступів і узгоджену колірну палітру. Темна тема підтримується

через клас `dark` на кореновому елементі; перемикач відповідно зберігає вибір у `localStorage`.

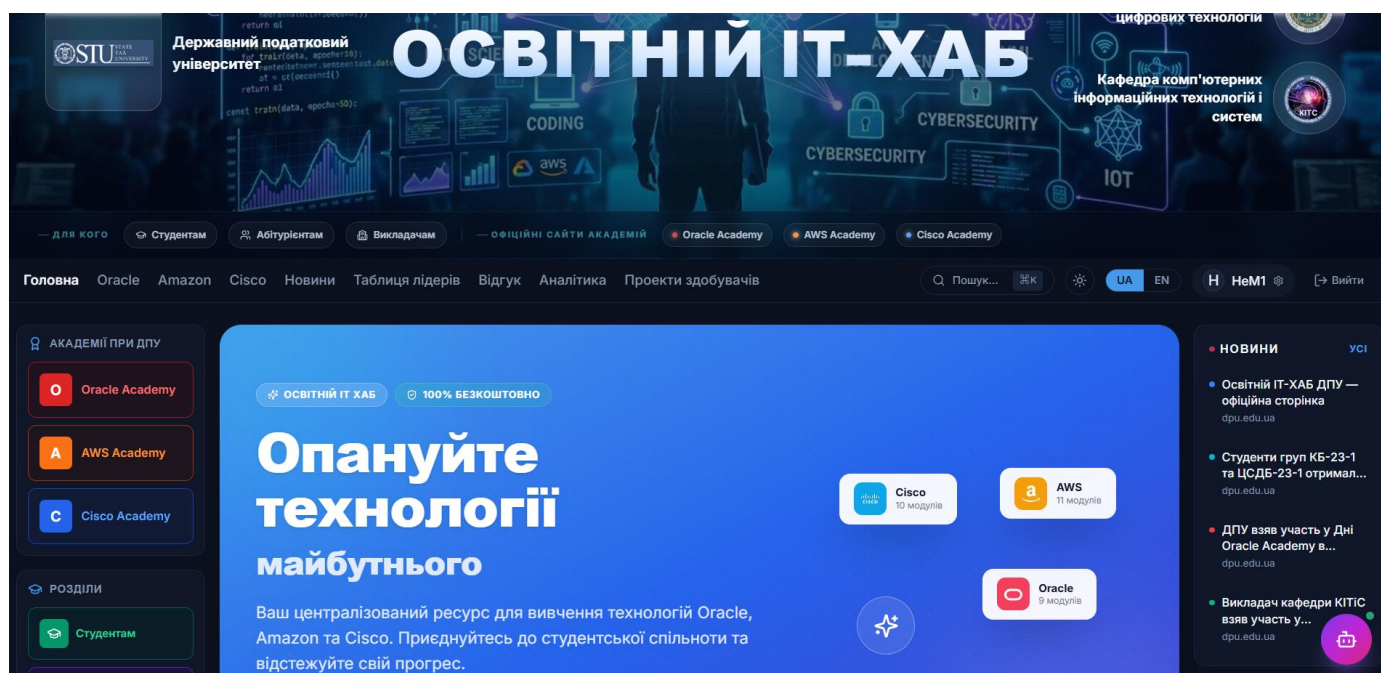


Рисунок 3.4 — (Головна сторінка)

Головна сторінка є точкою входу для всіх категорій відвідувачів. Вона складається з дванадцяти послідовних секцій: героїчний банер із градієнтним фоном (синьо-індигова гама) з плаваючими чіпами академій, три тематичних картки для цільових аудиторій, чотири анімованих лічильники статистики, картка прогресу авторизованого користувача, три картки академій, шість переваг платформи, покрокова інструкція з таймлайном, блок спільноти з опитуванням та топ-5, календар подій, стрічка партнерів і фінальний заклик до дії (рис. 3.4). Перемикач UA/EN у правому куті забезпечує миттєве перемикання мови без перезавантаження.

Із технічного погляду головна сторінка є найскладнішою: вона одночасно виконує п'ять паралельних запитів (статистика, прогрес, академії, опитування, таблиця лідерів) і відображає результати по мірі їх надходження. TanStack Query керує кешуванням кожного запиту незалежно: якщо дані про академії вже є у кеші, сторінка не чекатиме їх повторного завантаження. Це помітно прискорює суб'єктивну швидкість навігації між сторінками.

Секція «Живі лічильники» відображає чотири показники: загальну кількість зареєстрованих користувачів, кількість тематичних модулів, кількість пройдених тестів і відсоток безкоштовного доступу (100%). Анімація від нуля до цільового значення запускається лише тоді, коли елемент потрапляє у видиму область екрана — за допомогою хука `useInView` з `Framer Motion`. Завдяки цьому користувач, який повільно прокручує сторінку, завжди бачить повний цикл анімації, а не «застиглі» цифри.

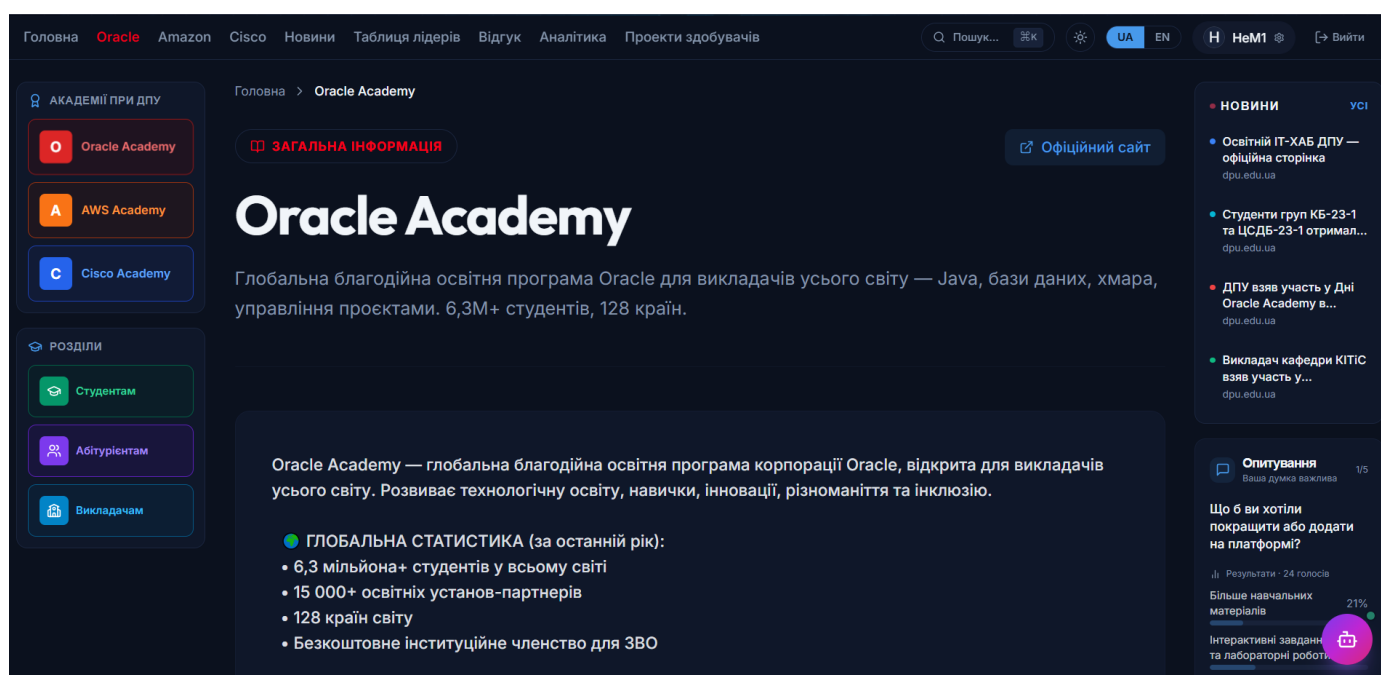


Рисунок 3.5— (Сторінка однієї з академій)

Кожна з трьох академій (Oracle Academy, AWS Academy, Cisco Networking Academy) має власну сторінку деталей (рис. 3.5). Сторінка містить: заголовок академії з логотипом та коротким описом, бейдж із поточним статусом проходження («Завершено» / «Продовжити» / «Розпочати»), нумерований список підтем із двомовними назвами та необов'язковими посиланнями на зовнішні матеріали, кнопку запуску тесту. Після успішного проходження тесту картка підсвічується зеленим кольором і відображає результат у відсотках.

Підтеми кожної академії зберігаються у полі `subTopics` типу `JSONB` і мають структуру: двомовна назва (`titleUk` / `titleEn`), двомовний опис (`descUk` / `descEn`) та необов'язкове

поле url із посиланням на зовнішній ресурс. Якщо url вказано, поруч із назвою підтеми з'являється кнопка «Детальніше», що відкривається у новій вкладці. Такий формат дозволяє додавати нові підтеми або редагувати наявні без змін у вихідному коді — достатньо оновити запис у базі.

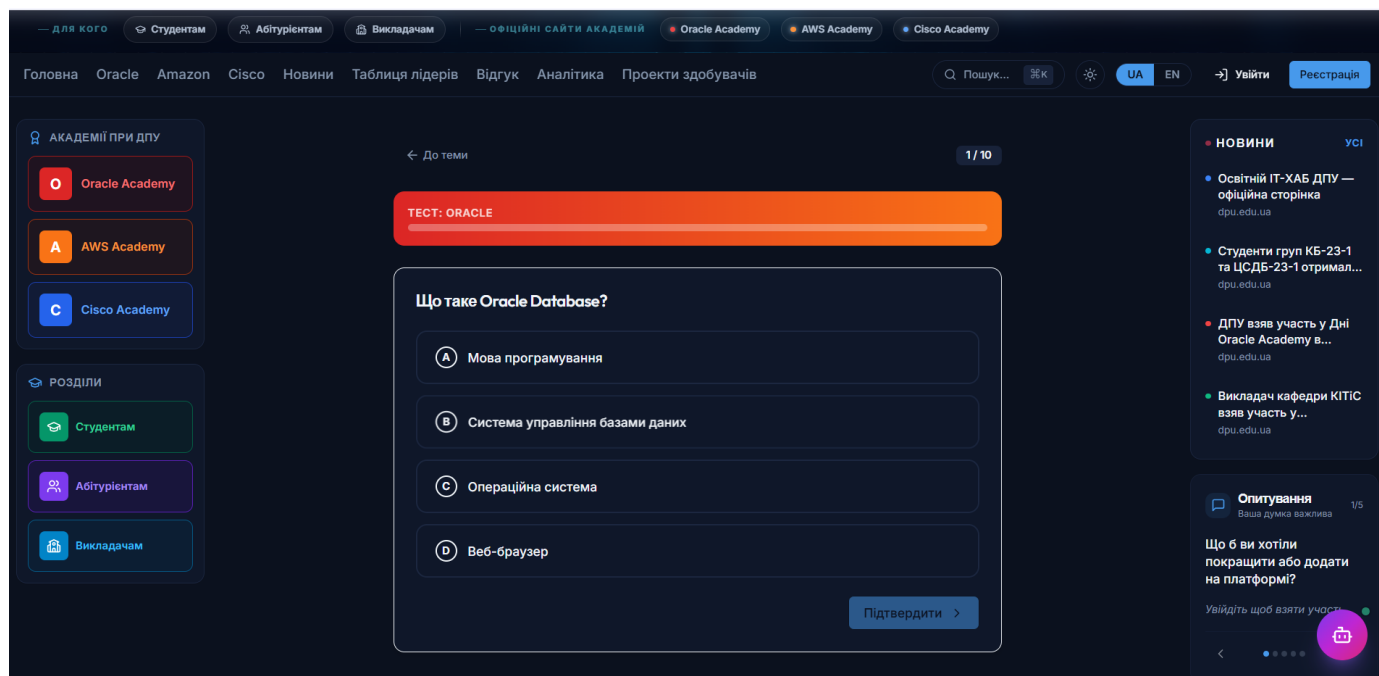


Рисунок 3.6 — (Тестування)

Сторінка тестування відображає питання по одному на екрані (рис. 3.6). Прогрес-бар у верхній частині наочно показує кількість питань, що залишилися. Варіанти відповідей оформлені у вигляді карток з hover-ефектом; після вибору картки блокуються і підсвічуються залежно від правильності (зелений — правильно, червоний — неправильно). Після останнього питання відображається підсумковий екран із відсотком правильних відповідей, анімованим колом прогресу та кнопками «Повторити» і «На головну».

Питання для тестів зберігаються безпосередньо у базі даних у вигляді JSONB-масиву у таблиці topics. Кожне питання має поля: текст питання, масив з чотирьох варіантів відповідей та індекс правильної відповіді. На клієнті питання перемішуються при кожному проходженні тесту — завдяки алгоритму Фішера–Єйтса, реалізованому у кілька рядків. Це запобігає механічному запам'ятовуванню порядку відповідей при повторних проходженнях.

Важливою деталлю є те, що правильна відповідь ніколи не надсилається клієнту заздалегідь — тільки після вибору варіанта. Це зроблено навмисно: якщо б масив питань містив індекс правильної відповіді з першого запиту, достатньо було б переглянути відповідь у Network-панелі DevTools, щоб дізнатись правильні варіанти наперед. Замість цього після кліку на варіант клієнт надсилає POST-запит із вибраним індексом, і сервер у відповіді повертає, чи він правильний, та пояснення.

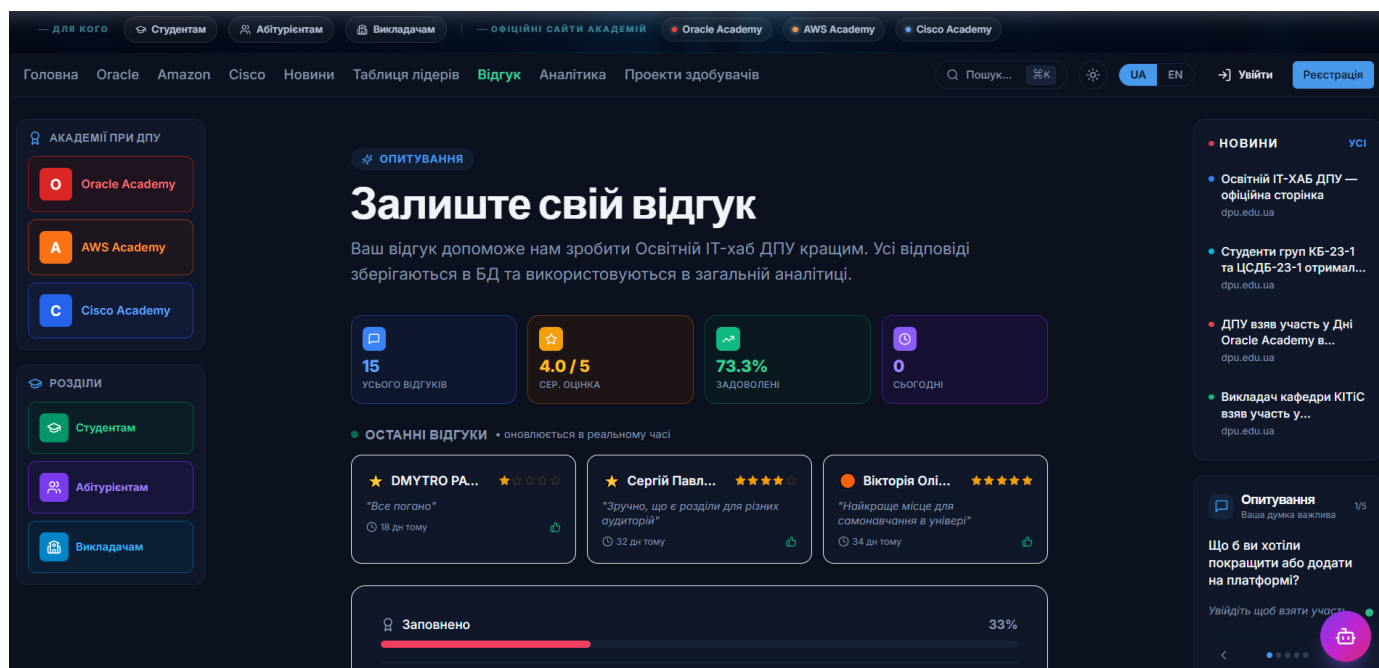


Рисунок 3.7 — (Сторінка зворотнього зв'язку)

Форма зворотного зв'язку поєднує кілька нестандартних елементів інтерфейсу (рис. 3.7). Вибір академії реалізовано через картки-перемикачі з логотипами; зіркові рейтинги за чотирма критеріями виводяться в один рядок; NPS-шкала 0–10 забарвлена градієнтом від червоного (0–6) через жовтий (7–8) до зеленого (9–10); перемикач «Подобається / Не подобається» умовно показує або приховує текстові поля. Валідація відбувається миттєво — підказки відображаються безпосередньо під полями без надсилення форми на сервер.

Вибір поля «Академія» реалізовано через картки-перемикачі, а не стандартний select-елемент. Кожна картка містить логотип академії, скорочену назву та індикатор вибору у вигляді кольорового обідка. Такий підхід є інтуїтивнішим для мобільних

користувачів — торкнутись великої картки значно зручніше, ніж розкрити дропдаун і знайти потрібний пункт.

Умовні текстові поля для позитивних і негативних аспектів з'являються або зникають залежно від стану перемикача «Подобається/Не подобається». Технічно це реалізовано через умовний рендер у JSX: `{liked && <textarea ... />}`. При перемиканні React видаляє елемент із DOM, тому дані введені у прихованому полі не надсилаються на сервер — що є правильною поведінкою з точки зору UX.



Рисунок 3.8 — (Сторінка аналітики)

Аналітичний дашборд є найнасиченішою за обсягом відображуваних даних сторінкою системи (рис. 3.8). У верхній частині розміщено панель фільтрів (7 днів / 30 днів / весь час, вибір академії) та п'ять KPI-карток. Нижче розташовано шість аналітичних блоків: кругова діаграма NPS, радарна діаграма за критеріями (контент / викладачі / платформа), лінійний графік часового ряду, стовпчаста діаграма порівняння академій, дві хмари слів та сортована таблиця відгуків. Кнопка «Експорт PDF» зберігає дашборд у файл без перезавантаження сторінки. Дані оновлюються автоматично кожні 30 секунд.

Хмара слів є нестандартним компонентом, написаним без сторонніх SVG-бібліотек. Кожне слово відображається як `span` із розміром шрифту, пропорційним частоті зустрічання (від 12 до 28 пікселів), і кольором із заздалегідь визначеної палітри. Перед

побудовою хмари сервер фільтрує стоп-слова — найчастіші службові слова обох мов (прийменники, займенники, артиклі), щоб у результаті залишились лише змістовні терміни.

Адміністративний модуль дашборду дозволяє завантажити усі відгуки у форматі CSV одним кліком. Файл генерується на сервері з UTF-8 BOM на початку — це необхідна деталь, без якої Microsoft Excel автоматично інтерпретує кодування як ANSI і показує кириличні символи як нечитабельний набір символів. Після завантаження адміністратор може відкрити файл у Excel або Google Sheets і самостійно будувати зведені таблиці.

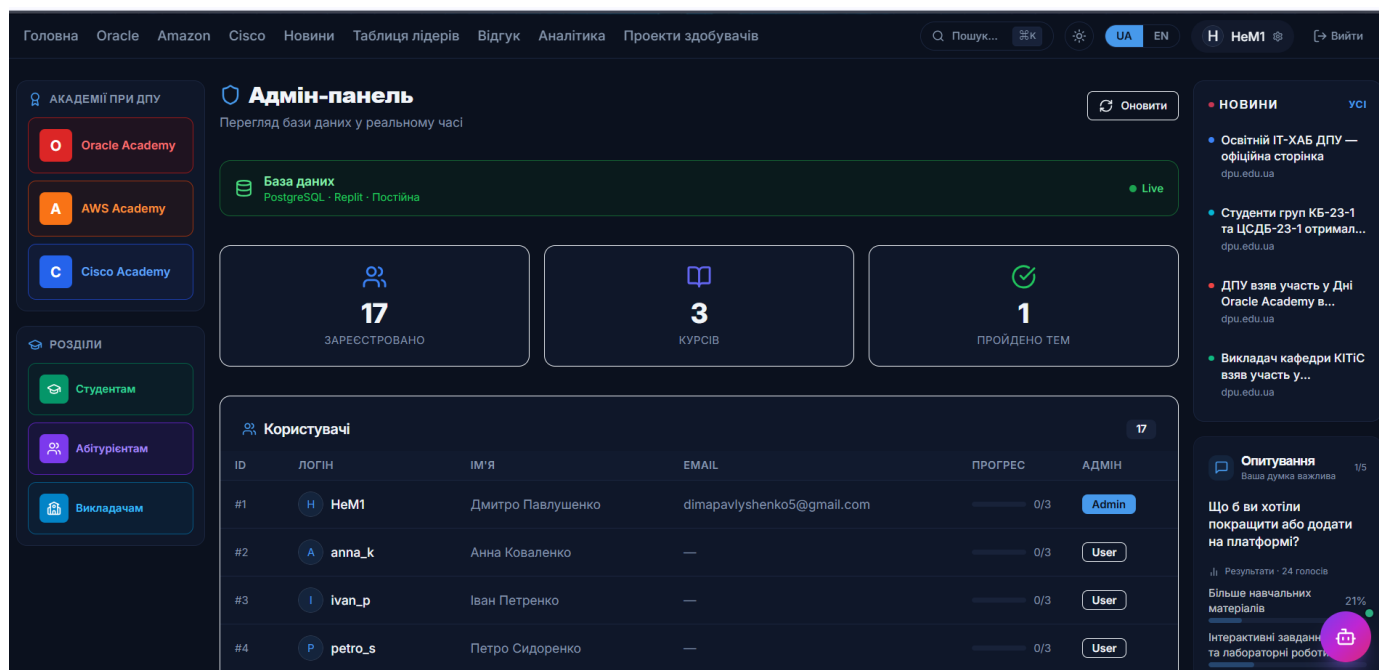


Рисунок 3.9 (Адмін-Панель)

Адміністративна панель (рис. 3.9), доступна лише адміністраторам, містить таблицю всіх відгуків, кнопки «Завантажити CSV» і «Очистити відгуки», а також аналітику поточних опитувань. Всі деструктивні операції захищено модальним діалогом підтвердження.

Бейджі досягнень обчислюються серверним методом `computeAchievements`, що приймає об'єкт із даними користувача (кількість завершених тем, наявність відгуку, участь в опитуванні, результати тестів, позиція в рейтингу) і повертає масив із 10 об'єктів. Кожен об'єкт містить булевий прапорець `earned`, двомовні назву й опис та

іконку. На клієнті бейджи з `earned: false` відображаються сірими із зменшеною прозорістю — щоб користувач бачив, що ще можна отримати, і це мотивувало до подальших дій.

Рекомендаційна система реалізована за простим, але ефективним алгоритмом: із трьох академій відфільтровуються ті, що ще не завершені поточним користувачем, і сортуються за пріоритетом (Oracle → AWS → Cisco — логічна послідовність від основ баз даних до хмарних технологій і мереж). До кожної рекомендації додається текстове пояснення двома мовами, чому саме ця академія рекомендується наступною. Максимальна кількість рекомендацій — три, щоб не перевантажувати користувача вибором.

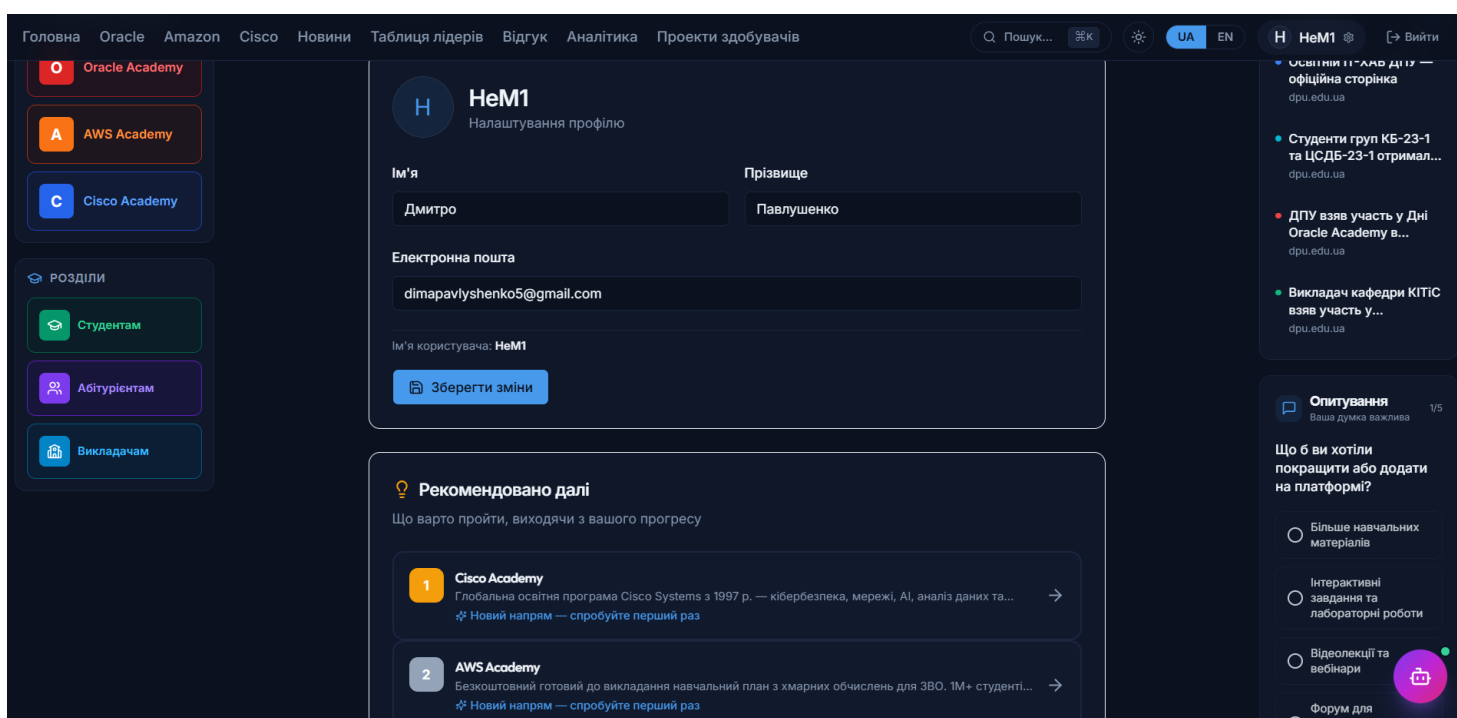


Рисунок 3.10 (Профіль користувача)

Сторінка особистого кабінету містить два блоки (рис. 3.10): сітку з 10 бейджів досягнень (здобуті — яскраво, ще не здобуті — сірим) та блок персоналізованих рекомендацій — до трьох карток із назвою академії та обґрунтуванням.

ВИСНОВКИ

У ході дипломної роботи вдалося довести до робочого стану інтерактивний освітній IT-портал кафедри КТіС факультету фінансів та цифрових технологій Державного податкового університету. Нижче коротко підсумовано, що саме було зроблено та які результати отримано.

Перш за все, було розглянуто чотири популярні платформи для онлайн- навчання — Moodle, Google Classroom, Coursera та GitHub Education — і порівняно їх між собою. Жодна з них не закриває одночасно всіх потреб кафедри з кількома навчальними програмами: немає прив'язки до трьох академій (Cisco, Oracle, AWS), відсутні власна аналітика за відгуками, гнучкі опитування та модулі під специфіку курсів КТіС. Саме цей висновок став обґрунтуванням того, щоб розробляти власне рішення замість використання готового продукту.

Аналіз відкрив цікаву закономірність: платформи, що «вміють усе», часто роблять це посередньо в усьому. Moodle технічно підтримує тести, рейтинги й форуми, але кожна з цих функцій виглядає як окремий «плагін», а не як органічна частина єдиного продукту. Це формує відчуття «клаптикової ковдри», а не цілісного навчального середовища. Для кафедри з трьома академіями і специфічними потребами власна розробка виявилась не просто кращим варіантом, а єдино правильним.

Технологічний стек підбирався так, щоб обидві частини застосунку працювали на одній мові. Тому і клієнт, і сервер написано на TypeScript. На клієнті використано React разом із Vite, маршрутизацію побудовано на Wouter, а робота з даними — на TanStack Query. Серверну частину реалізовано на Express.js із Drizzle ORM для доступу до бази та Passport.js для авторизації. Як сховище даних обрано PostgreSQL у хмарі Neon. Розгортання здійснюється через Replit Deployments — це дало змогу швидко отримати робочі версії ще на ранніх етапах розробки і не витратити час на ручне налаштування серверів.

Спроековано реляційну базу даних, що складається з шести основних таблиць: users, topics, progress, feedback_responses, polls і poll_responses. У схемі окремо продумано видалення нерелевантних або спам-записів так, щоб аналітика від

цього не страждала і зберігала свою цінність. Опис схеми виконано засобами Drizzle ORM, тому один і той самий файл одночасно слугує описом структури бази та джерелом TypeScript-типів для серверної логіки.

Особливою гордістю є рішення зберігати підтеми академій як JSONB, а не виносити їх в окрему таблицю. Для незмінного контенту це дало простоту запитів: замість двох JOIN-запитів — один SELECT. Водночас від початку структура схеми у `shared/schema.ts` слугує єдиним джерелом правди для TypeScript-типів як на фронтенді, так і на бекенді. Це суттєво зменшує кількість помилок типізації, які зазвичай виникають при дублюванні інтерфейсів.

Створено клієнтську частину, що складається з 14 сторінок, і серверну частину, у якій реалізовано 19 REST-ендпоінтів. Особливу увагу приділено інтерактивним функціям: тестам для перевірки знань, опитуванням із захистом від повторного голосування, формі зворотного зв'язку зі шкалою NPS, а також дашборду аналітики, який автоматично оновлюється у фоновому режимі без перезавантаження сторінки.

Найбільш трудомісткою виявилась розробка дашборду аналітики — не через складність самих графіків, а через необхідність правильно агрегувати різноманітні дані в одному запиті та повертати структуровану відповідь, якій довіряють усі клієнтські компоненти. Загалом розробка аналітичного ендпоінту зайняла більше часу, ніж усі інші ендпоінти разом.

Систему протестовано за 11 основними сценаріями використання. О. М. Спірін зазначає, що «якість ІКТ-системи навчання доцільно оцінювати за критеріями функціональності, надійності та зручності використання» [37]., а також у граничних і свідомо некоректних випадках. У всіх перевірках платформа поведилась передбачувано: дані зберігалися без втрат, у відповідь на помилки повертались відповідні HTTP-коди, а цілісність бази даних не порушувалась.

Готову платформу розгорнуто і запущено в реальну експлуатацію за адресою `dpu-hub-2--dimapavlyshenko.replit.app`. У подальшому планується додати автоматизоване тестування на рівні CI, верифікацію поштових адрес користувачів,

підключення нових партнерських академій, а також перенесення проєкту на офіційний домен ДПУ.

Загалом виконана робота показала, що навіть невеликою командою (а в даному випадку — самотійно) можна побудувати повноцінне освітнє веб-середовище для конкретної кафедри, спираючись на сучасні безкоштовні інструменти та хмарні сервіси. Отриманий портал уже використовується для роботи зі студентами і має простір для подальшого розширення.

Підсумовуючи, можна стверджувати, що мета роботи досягнута повністю: розроблено і розгорнуто функціональний освітній IT-портал, що об'єднує три академічні програми в єдиному інтерфейсі, підтримує двомовність, гейміфікацію, систему відгуків з аналітикою та адміністративний модуль. Портал вже розгорнуто на Replit за постійною адресою і готовий до реального використання студентами та викладачами кафедри ДПУ.

Практична значимість роботи полягає не лише у готовому продукті, а й у набутих навичках: проектування реляційної бази даних, побудова типобезпечного full-stack застосунку на TypeScript, реалізація сесійної автентифікації, інтеграція аналітичних бібліотек та налагодження хмарного розгортання. Ці компетенції безпосередньо затребувані на ринку праці для розробників ПЗ і слугуватимуть міцним фундаментом для подальшого розвитку у напрямку веб-розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Биков В. Ю. Моделі організаційних систем відкритої освіти : монографія. Київ : Атіка, 2009. URL: https://lib.iitta.gov.ua/id/eprint/845/1/Bykov_Models.pdf
2. Морзе Н. В., Глазунова О. Г. Критерії якості електронних навчальних курсів, розроблених на базі платформ дистанційного навчання. Інформаційні технології в освіті
URL: <https://www.journal.iitta.gov.ua/index.php/itlt/article/view/5032>
3. Триус Ю. В., Герасименко І. В. Комбіноване навчання як інноваційна освітня технологія у вищій школі. Теорія та методика електронного навчання. Кривий Ріг, 2012. URL: <https://surl.li/cfjcux>
4. Спірін О. М. Інформаційно-комунікаційні та інформатичні компетентності як компоненти системи професійно-спеціалізованих компетентностей вчителя інформатики. Інформаційні технології і засоби навчання. 2009. № 5(13). URL: <https://surl.li/rbsxgc>
5. Кухаренко В. М., Бондаренко В. В. та ін. Екстрене дистанційне навчання в Україні: монографія / за ред. В. М. Кухаренка. Харків : КП «Міська друкарня», 2020. URL: <https://surl.li/lehzox>
6. Про вищу освіту: Закон України від 01.07.2014 № 1556-VII. URL: <https://zakon.rada.gov.ua/laws/show/1556-18>
7. Шишкіна М. П. Формування і розвиток хмаро орієнтованого освітньо-наукового середовища вищого навчального закладу : монографія. Київ : УкрІНТЕІ, 2015. URL: <https://lib.iitta.gov.ua/id/eprint/166208/1/monog-e-lib.pdf>
8. Биков В. Ю., Спірін О. М., Пінчук О. П. Сучасні завдання цифрової трансформації освіти. Вісник Кафедри ЮНЕСКО «Неперервна професійна освіта XXI століття». 2020. URL: <https://surl.li/zuwgyc>
9. Концепція розвитку цифрової економіки та суспільства України на 2018–2020 роки: Розпорядження Кабінету Міністрів України від 17.01.2018 № 67-р. URL: <https://zakon.rada.gov.ua/laws/show/67-2018-p>

10. Буйницька О. П. Інформаційні технології та технічні засоби навчання : навч. посіб. Київ : Центр учбової літератури. URL: <https://surl.li/vouslm>

11. Гуревич Р. С., Кадемія М. Ю. Інформаційно-телекомунікаційні технології в навчальному процесі та наукових дослідженнях : навч. посіб. Вінниця : ДОВ «Вінниця». URL: <https://surl.li/vztery>

12. Триус Ю. В. Комп'ютерно-орієнтовані методичні системи навчання математики : монографія. Черкаси : Брама-Україна. URL: <https://surl.li/ltzibf>

13. Кухаренко В. М., Рибалко О. В., Сиротенко Н. Г. Дистанційне навчання: умови застосування : навч. посіб. / за ред. В. М. Кухаренка. Харків : НТУ «ХПІ». URL: <https://surl.li/oxttau>

14. Дичківська І. М. Інноваційні педагогічні технології : підручник. 3-тє вид., доп. Київ : Академвидав. URL: <https://surl.li/kljujk>

15. Сікора Я. Б. Розвиток компетентності з програмування майбутніх учителів інформатики засобами мережевих технологій. URL: https://eprints.zu.edu.ua/5558/2/Sikora_Vin.pdf

16. Глазунова О. Г., Морзе Н. В. Проектування системи електронного навчання для вищого навчального закладу : навч.-метод. посіб. Київ. URL: <https://surl.li/yvtfin>

17. Осадча К. П. Педагогічні засади проектування систем дистанційного навчання майбутніх учителів : дис. ... канд. пед. Наук. URL: https://eprints.mdpu.org.ua/id/eprint/11532/1/dis_OsadchaKP_.pdf

18. Сороко Н. В. Розвиток інформаційно-комунікаційної компетентності вчителів філологічної спеціальності в умовах комп'ютерно орієнтованого середовища : автореф. дис. ... канд. пед. Наук. URL: https://lib.iitta.gov.ua/id/eprint/717978/1/Posibn_Ovcharuk-1.pdf

19. Вакалюк Т. А. Теоретико-методичні засади застосування хмарних технологій у підготовці бакалаврів інформатики : дис. ... д-ра пед. Наук. URL: <https://surl.li/mjlqon>
20. Литвинова С. Г. Теоретико-методичні основи проектування хмаро орієнтованого навчального середовища загальноосвітнього навчального закладу : дис. ... д-ра пед. наук : URL: <https://surl.li/ikmxyn>
21. Семеріков С. О., Теплицький І. О., Словак К. І. Нові засоби мобільного навчання у вищій школі // Вища освіта України. 2010. URL: <https://surl.li/vneivj>
22. Стрюк А. М., Семеріков С. О. Мобільність: системний підхід // Інформаційні технології і засоби навчання. URL: <https://surl.li/azyrrx>
23. Морзе Н. В., Кочарян А. Б. Модель стандарту ІКТ-компетентності викладачів університету в контексті підвищення якості освіти // Інформаційні технології і засоби навчання. 2014. URL: <https://surl.li/omejzb>
24. Литвинова С. Г. Хмаро орієнтоване навчальне середовище загальноосвітнього навчального закладу: від терміну до моделі // Інформаційні технології і засоби навчання. 2014. URL: <https://ceur-ws.org/Vol-2168/paper2.pdf>
25. Вакалюк Т. А. Переваги використання хмарних технологій у навчальному процесі вищих навчальних закладів // Педагогіка вищої та середньої школи. 2014. URL: <https://surl.li/dgdkxe>
26. Гриценчук О. О., Іванова С. М., Кравчина О. Є. та ін. Рамкова програма цифрової компетентності DigComp 2.0: ключові концептуальні моделі // Інформаційні технології і засоби навчання. 2019. URL: <https://surl.li/vogyjo>
27. Буйницька О. П., Варченко-Троценко Л. О. Використання систем управління навчанням у вищій освіті: аналіз функціонального потенціалу // Відкрите освітнє е-середовище сучасного університету. 2015. URL: <https://surl.li/hirjua>

28. Шишкіна М. П., Попель М. В. Хмаро орієнтоване освітнє середовище навчального закладу: сучасний стан і перспективи розвитку // Інформаційні технології і засоби навчання. 2013. URL: <https://surli.cc/vjxwva>
29. Нечипуренко П. П., Старова Т. В., Семеріков С. О. Використання хмаро орієнтованих лабораторних комплексів у природничо-науковій освіті старшокласників // Наукові записки. URL: <https://surl.li/oyqewl>
30. Попель М. В. SageMathCloud як засіб навчання вищої математики студентів вищих навчальних закладів // Інформаційні технології і засоби навчання. URL: <https://lib.iitta.gov.ua/id/eprint/704647/1/148-165.pdf>
31. Яцишин А. В., Коваленко В. В. Застосування відкритих електронних науково-освітніх систем для розвитку інформаційно-дослідницької компетентності наукових і науково-педагогічних працівників // Інформаційні технології і засоби навчання. URL: <https://surl.li/pfubid>
32. Сисоєва С. О., Осадча К. П. Зарубіжний досвід дистанційного навчання у вищій освіті // Педагогічна освіта: теорія і практика. Психологія. Педагогіка. URL: <https://surl.li/engamv>
33. Биков В. Ю., Задорожна Н. Т. Концептуальні засади забезпечення відкритого навчального середовища наукових установ НАПН України // Інформаційно-комунікаційні технології в сучасній освіті: досвід, проблеми, перспективи. Львів : ЛДУ БЖД, URL: <https://surl.li/arjtos>
34. Семеріков С. О., Мінтій І. С., Словак К. І. Мобільне математичне навчання: теорія та практика // Математика. Комп'ютер. Освіта : матеріали XIX міжнар. конф. (Пущіно, 2012). URL: <https://elibrary.kdpu.edu.ua/xmlui/handle/0564/929>
35. Литвинова С. Г., Буров О. Ю. Методи, форми та електронні засоби дистанційного навчання у процесі підготовки майбутніх учителів // Науковий часопис НПУ імені М. П. Драгоманова. 2017. URL: <https://surl.li/uzyqxc>

36. Дем'яненко В. М., Шишкіна М. П. Проблеми підготовки педагогічних кадрів до використання хмарних сервісів // Нові інформаційні технології в освіті для всіх. URL: <https://surl.li/emtedv>
37. Карташова Л. А., Березенська С. М. Реалізація сервісів хмарних обчислень при підготовці вчителів інформатики // Освіта та розвиток обдарованої особистості. URL: <https://surl.lu/sspjwa>
38. Іванова С. М. Наукова електронна бібліотека НАПН України як засіб інформаційно-аналітичної підтримки педагогічних досліджень // Інформаційні технології і засоби навчання. URL: <https://surl.li/usjnma>
39. Переваги використання онлайн-середовища розробки «replit» для вивчення мови програмування «python» у закладах освіти під час дистанційного навчання. URL: <https://surl.lu/jzudom>
40. Павлушенко Д. М. Приклад розробки REST API та тестування системи на прикладі освітнього веб-порталу. Інноваційні цифрові технології: від алгоритмів до безпеки : матеріали наук.-практ. конф. (Ірпінь, травень 2026 р.). Ірпінь : ДПУ, 2026. С. 115–118.