ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-платформи для надання доступу до освітніх IT-курсів»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Вадим МОЙСЕЄНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Вадим МОЙСЕЄНКО

Керівник: Владислав ЯСКЕВИЧ
к.т.н., доцент

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мойсеєнку Вадиму Денисовичу

1. Тема кваліфікаційної роботи: «Розробка web-платформи для надання доступу до освітніх ІТ-курсів»
керівник кваліфікаційної роботи к.т.н., доцент Владислав ЯСКЕВИЧ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про принципи розробки web-платформ, опис існуючих освітніх платформ для доступу до ІТ-курсів, аналіз особливостей організації онлайн-навчання в сфері ІТ, огляд сучасних технологій розробки web-додатків та технічна документація до використаних фреймворків та бібліотек.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз існуючих онлайн-платформ для проходження ІТ-курсів.
2. Проєктування web-платформи для надання доступу до освітніх ІТ-курсів.
3. Програмна реалізація та опис функціонування web-платформи для надання доступу до освітніх ІТ-курсів.

4. Тестування платформи.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до web-платформи.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Структура бази даних.
6. Мапи web-платформи.
7. Екранні форми.
8. Апробація результатів дослідження.

6. Дата видачі завдання«25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Огляд існуючих онлайн-платформ для проходження IT-курсів	14.03-19.03.2025	
4	Проєктування web-платформи для надання доступу до освітніх IT-курсів	20.03-27.03.2025	
5	Програмна реалізація платформи	28.03-24.04.2025	
6	Тестування платформи	25.04-28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Вадим МОЙСЕЄНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Владислав ЯСКЕВИЧ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 54 стор., 2 табл., 31 рис., 20 джерел.

Мета роботи – спрощення доступу до освітніх ІТ-курсів шляхом розробки web-платформи для створення, модерації та проходження онлайн-курсів.

Об'єкт дослідження – процес організації онлайн-навчання у сфері інформаційних технологій.

Предмет дослідження – web-платформа Codegram для створення та проходження онлайн-курсів з функціями модерації та відстеження прогресу.

Короткий зміст роботи: В роботі проаналізовано особливості організації онлайн-навчання в сфері ІТ. Проаналізовано аналоги освітніх онлайн-платформ: Prometheus, EdEra, Udemy, Coursera. Розроблено архітектуру web-платформи та програмно реалізовані ключові функціональні можливості, зокрема: реєстрація користувачів, створення курсів, модерація контенту, проходження навчальних модулів, відстеження прогресу навчання, складання тестів. Проведено тестування web-платформи за допомогою тестових сценаріїв. В роботі використано фреймворк Vue.js для створення користувацького інтерфейсу, Firebase для backend-функціональності, Pinia для управління станом додатку.

Сферою використання застосунку є організація онлайн-навчання та професійного розвитку у сфері інформаційних технологій.

КЛЮЧОВІ СЛОВА: WEB-ПЛАТФОРМА, ОСВІТНІ ІТ-КУРСИ, ОНЛАЙН-НАВЧАННЯ, VUE.JS, FIREBASE, PINIA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface

BaaS - Backend-as-a-Service

ES – ECMAScript

HTML - HyperText Markup Language

JS – JavaScript

VS Code - Visual Studio Code

ВСТУП

У сучасному світі інформаційних технологій освіта стає ключовим фактором успіху та кар'єрного зростання. Стрімкий розвиток IT-галузі створює постійну потребу в кваліфікованих фахівцях, що володіють актуальними знаннями та навичками. Традиційні методи навчання часто не встигають за швидкістю змін технологій, що зумовлює зростаючу популярність онлайн-освіти, яка забезпечує гнучкість, доступність та можливість своєчасного оновлення навчальних матеріалів відповідно до тенденцій галузі.

Мета роботи – спрощення доступу до освітніх IT-курсів шляхом розробки web-платформи для створення, модерації та проходження онлайн-курсів.

Об'єкт дослідження – процес організації онлайн-навчання у сфері інформаційних технологій.

Предмет дослідження – web-платформа Codegram для створення та проходження онлайн-курсів з функціями модерації та відстеження прогресу.

Для реалізації поставленої мети виділено такі задачі:

1. Провести аналіз існуючих онлайн-платформ для проходження IT-курсів, виявити їхні переваги та недоліки.
2. Дослідити сучасні технології та інструменти, які можуть бути використані для розробки web-платформи.
3. Сформулювати перелік функціональних та нефункціональних вимог до майбутньої платформи.
4. Спроєктувати архітектуру web-платформи для доступу до освітніх IT-курсів, що забезпечує функціонал створення, модерації та проходження онлайн-курсів.
5. Реалізувати програмну частину платформи відповідно до визначених вимог та спроєктованої архітектури.
6. Провести тестування розробленої системи.

Деякі результати дослідження доповідались на студентських наукових конференціях: VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» за темою «Порівняльний аналіз фреймворків vue.js та react у сучасній веброзробці» [1]; IV Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT» за темою «Переваги та недоліки використання бази даних firestore для розробки вебдодатків» [2].

Для розробки web-платформи для освітніх IT-курсів було обрано сучасний JavaScript-фреймворк Vue.js, який вирізняється легкістю освоєння та високою продуктивністю завдяки компонентній архітектурі та реактивності. Для управління станом додатку використовується бібліотека Pinia, що надає підтримку типізації та модульність. Маршрутизація реалізована за допомогою Vue Router, що дозволяє створювати динамічні маршрути та захищені області для авторизованих користувачів. В якості бекенду обрано Firebase – комплексне рішення від Google, що надає широкий спектр інструментів для розробки web-додатків, включаючи Firestore як NoSQL базу даних та Authentication для управління користувачами. Використання Firebase дозволяє значно прискорити розробку платформи без необхідності розгортання власної серверної інфраструктури.

Результатом роботи є web-платформа Codegram, що надає користувачам можливість створювати освітні IT-курси, проходити їх та відстежувати свій прогрес. Завдяки системі модерації користувачі отримують доступ до якісних навчальних матеріалів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості організації онлайн-навчання в сфері ІТ

Освіта в сфері інформаційних технологій є одним із найбільш динамічних напрямків сучасної освіти, що стикається з унікальними викликами та можливостями. Ця галузь характеризується надзвичайно швидким розвитком технологій, постійною зміною вимог до фахівців та необхідністю адаптації навчальних програм до реальних потреб ринку праці.

Стрімкий технологічний розвиток та постійне розширення вимог до ІТ-спеціалістів створює потребу в нових підходах до освіти, оскільки традиційні методи навчання часто не встигають за швидкоплинними змінами на ринку праці [3, с. 124]. Традиційні заклади освіти, зокрема університети та коледжі, стикаються з труднощами в оперативному оновленні навчальних програм, які нерідко втрачають актуальність ще до завершення студентами навчання. У результаті виникає розрив між теоретичними знаннями, що надаються в межах академічної підготовки, та практичними навичками, необхідними для успішної професійної діяльності в ІТ-сфері.

Онлайн-освіта, як невід'ємна частина сучасного освітнього простору, пропонує студентам суттєві конкурентні переваги через розвиток затребуваних роботодавцями навичок самоорганізації, відповідальності та аналітичних здібностей [4, с. 52]. Онлайн-курси дозволяють навчатися у зручний час, поєднувати навчання з роботою та отримувати актуальні знання. Крім того, вони надають можливість швидко реагувати на зміни в технологіях та підходах, включаючи нові теми та інструменти в навчальні програми.

Цифрова трансформація освіти вимагає не лише створення відповідних електронних ресурсів та сервісів, але й формування інноваційної компетентності, що передбачає відкритість до нових ідей та здатність ініціювати зміни у своєму середовищі [5, с. 102]. Така трансформація змінює саму

парадигму навчання, переходячи від передачі фіксованого набору знань до формування здатності постійно навчатися та адаптуватися до нових технологій.

Впровадження інноваційних технологій значно прискорило розвиток онлайн-навчання, змінюючи традиційний освітній процес. Пандемія COVID-19 та війна в Україні посилили цю тенденцію, підкресливши необхідність відкритих ресурсів та онлайн-курсів, які дозволяють здобувати якісну освіту незалежно від місця перебування та формують важливі навички самоорганізації та самостійного засвоєння знань [6].

Основними користувачами таких платформ є студенти, які прагнуть здобути актуальні практичні навички та викладачі, які шукають ефективні інструменти для створення та адаптації навчального контенту. Для таких користувачів важливою є наявність інструментів, які забезпечують зручність навчання, актуальність змісту та можливість самостійно контролювати темп засвоєння матеріалу.

Таким чином, в умовах постійних змін та високої динаміки IT-сектору, актуальним є створення спеціалізованої web-платформи, яка б забезпечувала повний цикл взаємодії: від розробки та публікації онлайн-курсів до їх проходження з відстеженням результатів навчання. Такий інструмент сприятиме підвищенню якості освіти, задоволенню потреб усіх учасників освітнього процесу та дозволить адаптувати процес навчання до індивідуальних потреб користувачів в умовах стрімких технологічних змін.

1.2 Аналіз аналогів

Для розробки ефективної web-платформи для проходження освітніх IT-курсів необхідно провести детальний аналіз існуючих рішень, що вже функціонують у цій сфері. Сучасний ринок онлайн-освіти представлений великою кількістю платформ, які надають доступ до навчального контенту, зокрема у сфері інформаційних технологій. Кожен із цих сервісів має свої особливості, переваги та недоліки, що стосуються організації навчального

процесу, інтерактивності, актуальності матеріалів та способу представлення змісту.

З огляду на це, доцільним є розгляд найпоширеніших освітніх платформ, що спеціалізуються на IT-напрямках. До таких платформ можна віднести:

- Prometheus;
- EdEra;
- Udemy;
- Coursera.

1.2.1 Prometheus

Prometheus – це найбільша онлайн-платформа професійного розвитку в Україні, що співпрацює з провідними українськими університетами та IT-компаніями [7]. Платформа надає структуровані курси з IT, бізнесу, підприємництва та громадянської освіти українською мовою (див. рисунок 1.1).

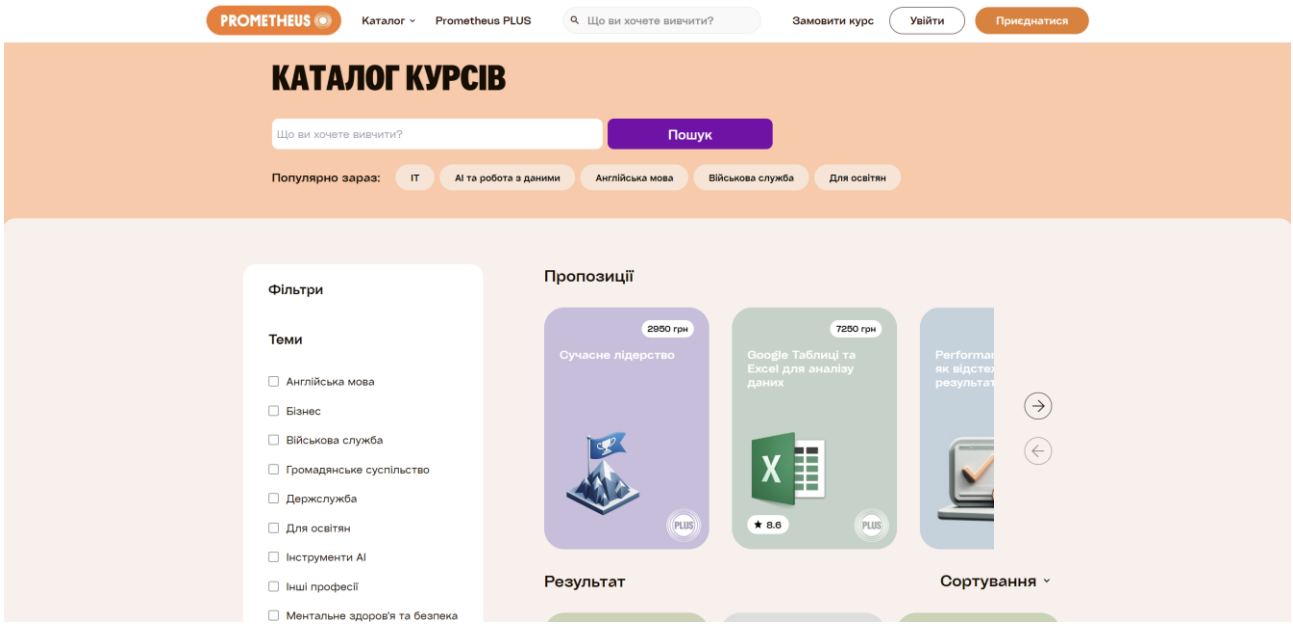


Рис. 1.1. Каталог курсів на платформі Prometheus

Переваги платформи:

- Безкоштовний доступ до більшості навчальних матеріалів;
- Курси від провідних викладачів та науковців;
- Україномовний інтерфейс;

- Підтримка мобільних пристроїв для навчання в будь-який час;
- Ефективна система відстеження прогресу.

Недоліки платформи:

- Обмежений вибір спеціалізованих IT-курсів;
- Відсутність можливості для користувачів створювати власні курси;
- Обмежена інтерактивність навчальних матеріалів.

Таким чином, Prometheus відіграє важливу роль у розвитку україномовної онлайн-освіти, проте має суттєві обмеження для повноцінного навчання в IT сфері через недостатню різноманітність напрямків та відсутність можливості для користувачів створювати власні курси.

1.2.2 EdEra

EdEra - українська студія онлайн-освіти, що створює інтерактивні навчальні курси та освітній контент [8]. Платформа відзначається сучасним підходом до подачі матеріалу з використанням мультимедійних елементів та інтерактивних завдань (див. рисунок 1.2).

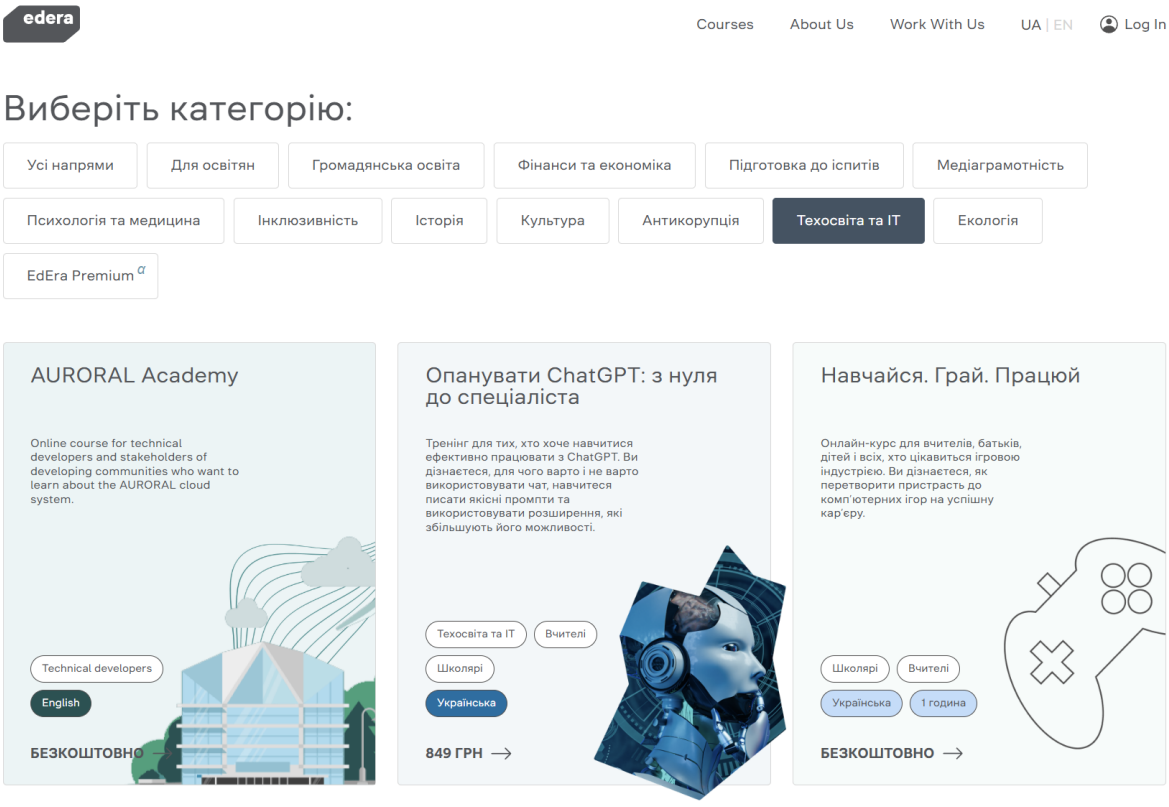


Рис. 1.2. Каталог IT курсів на платформі EdEra

Основні переваги платформи:

- Інтерактивні онлайн-курси з різних дисциплін, включаючи основи програмування;
- Україномовний інтерфейс;
- Мобільна версія для навчання у будь-який час та місці;
- Система відстеження прогресу проходження курсу.

Недоліки платформи:

- Фокус на загальній та шкільній освіті;
- Недостатня глибина для професійного навчання програмуванню;
- Відсутність можливості для користувачів створювати власний контент.

Таким чином, EdEra представляє якісний україномовний освітній контент, але має значні обмеження для професійного IT-навчання через недостатню кількість спеціалізованих курсів та відсутність поглибленого матеріалу для професійного зростання в IT-сфері.

1.2.3 Udemy

Udemy – це глобальний маркетплейс онлайн-курсів, де викладачі можуть створювати та публікувати власні навчальні матеріали, а студенти – купувати та проходити ці курси [9]. Платформа надає можливості для авторів курсів, які можуть самостійно визначати зміст, формат та вартість своїх матеріалів (див. рисунок 1.3).

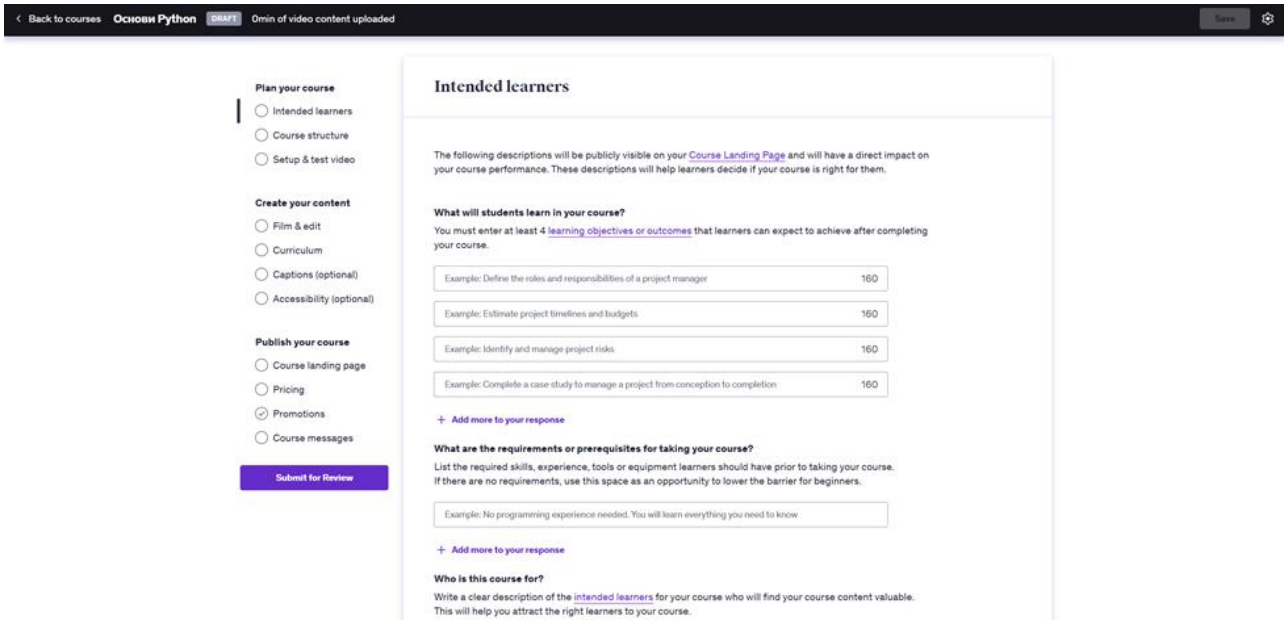


Рис. 1.3. Інтерфейс створення курсу на платформі Udemy

Основні переваги:

- Велика різноманітність IT-напрямків, що охоплює широкий спектр технологій;
- Можливість для користувачів створювати та публікувати власні навчальні матеріали;
- Система модерації користувацького контенту.

Недоліки:

- Відсутність україномовного інтерфейсу;
- Невелика кількість безкоштовних матеріалів.

Платформа Udemy вирізняється великою різноманітністю IT-курсів і надає можливість користувачам самостійно створювати навчальний контент, що сприяє швидкому оновленню та розширенню навчальної бази. Проте відсутність україномовного інтерфейсу та обмежений доступ до безкоштовних матеріалів можуть створити бар’єри для деяких користувачів.

1.2.4 Coursera

Coursera – це освітня платформа, що співпрацює з провідними університетами та компаніями для створення і розповсюдження онлайн-курсів [10]. Платформа пропонує структурований підхід до навчання з чітко визначеними програмами та шляхами розвитку навичок (див. рисунок 1.4).

Основні переваги платформи:

- Велика різноманітність IT-напрямків;
- Добре структуровані програми навчання;

Недоліки платформи:

- Висока вартість підписки або окремих курсів;
- Відсутність можливості створювати власний контент;
- Фіксований графік завершення для деяких курсів.

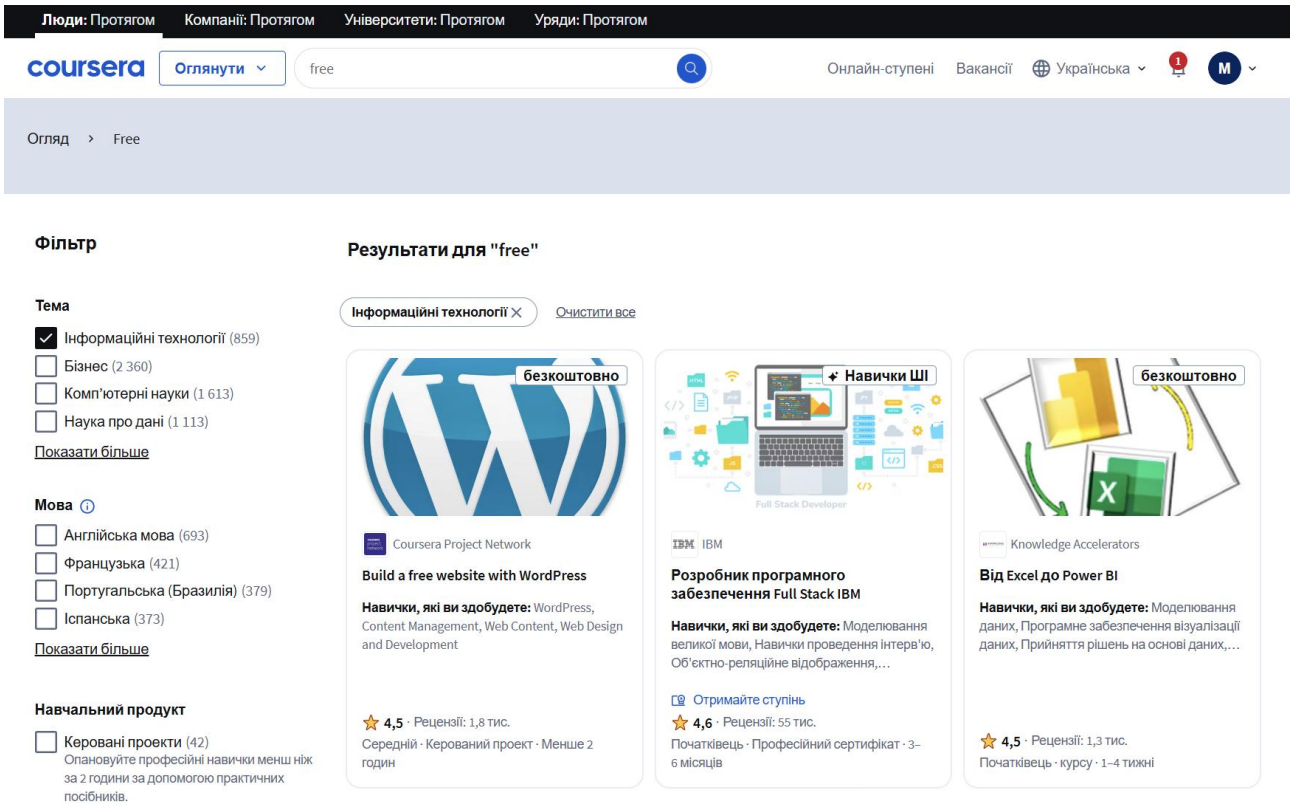


Рис. 1.4. Каталог IT курсів на платформі Coursera

Таким чином, Coursera надає доступ до якісного контенту з україномовним інтерфейсом, проте висока вартість, недостатня кількість українських матеріалів та відсутність можливості для створення власного контенту є вагомими факторами, що знижують доцільність її використання.

1.2.5 Таблиця порівнянь аналогів

Зведені результати аналізу характеристик розглянутих web-платформ наведено у таблиці 1.1.

Таблиця 1.1

Порівняння існуючих аналогів

Характеристика	Prometheus	EdEra	Udemy	Coursera	Codegram
Українська мова інтерфейсу	+	+	-	+	+
Велика різноманітність IT-напрямків	-	-	+	+	+
Можливість створення курсів користувачами	-	-	+	-	+
Підтримка мобільних пристроїв	+	+	+	+	+
Модерація користувацького контенту	-	-	+	-	+
Відстеження прогресу	+	+	+	+	+

2 ПРОЄКТУВАННЯ WEB-ПЛАТФОРМИ

2.1 Функціональні на нефункціональні вимоги

Дослідивши існуючі аналоги освітніх платформ та проаналізувавши потреби користувачів, було сформовано основні функціональні вимоги до розроблюваної web-платформи:

1. Реєстрація та авторизація користувачів. Система повинна надавати користувачам можливість створювати нові облікові записи та безпечно авторизуватися для доступу до функціоналу платформи.
2. Реалізація системи ролей із різними правами доступу. Платформа повинна розмежовувати функціональні можливості для різних типів користувачів (користувач, модератор), забезпечуючи контроль доступу до ресурсів системи.
3. Можливість пошуку та фільтрації курсів. Користувачі повинні мати можливість здійснювати пошук курсів за ключовими словами та фільтрувати їх за різними параметрами для зручного вибору потрібного навчального матеріалу.
4. Можливість проходження курсів користувачами з відстеженням прогресу. Платформа повинна забезпечувати функціонал для послідовного проходження навчальних матеріалів із автоматичним збереженням прогресу користувача.
5. Можливість створення, редагування та видалення курсів авторами. Модератори повинні мати інструменти для створення структурованих навчальних курсів із різними типами контенту та подальшого керування ними.
6. Можливість перегляду профілів та публікацій інших користувачів. Система повинна забезпечувати доступ до публічних профілів інших користувачів платформи та їхніх навчальних матеріалів.

7. Адміністративна панель для управління курсами. Платформа повинна надавати функціонал для ефективного керування навчальним контентом, користувачами та системними налаштуваннями.

Нефункціональні вимоги визначають якість web-платформи, впливаючи на зручність використання, надійність та ефективність її роботи. До даних вимог належать:

1. Захист паролів користувачів. Система повинна забезпечувати надійне зберігання паролів користувачів з використанням сучасних методів хешування для запобігання несанкціонованому доступу до облікових записів.

2. Адаптивний дизайн інтерфейсу для різних розмірів екрану. Платформа повинна коректно відображатися та зберігати функціональність на пристроях з різними розмірами екранів, забезпечуючи зручність використання на мобільних пристроях і десктопах.

3. Швидке завантаження сторінок сайту. Час завантаження сторінок платформи не повинен перевищувати 3 секунди для забезпечення позитивного користувацького досвіду.

4. Підтримка сучасних браузерів. Web-платформа повинна коректно функціонувати у всіх популярних сучасних браузерах (Chrome, Firefox, Edge, Opera), забезпечуючи однаковий користувацький досвід незалежно від обраного програмного забезпечення.

2.2 Проєктування варіантів використання

Варіанти використання описують способи взаємодії користувача з web-платформою для навчання програмуванню та розвитку технічних навичок з метою задоволення освітніх потреб. Діаграма варіантів використання є цінним інструментом для візуалізації цих взаємодій та розуміння функціональних вимог системи з точки зору різних типів користувачів (див. рисунок 2.1).

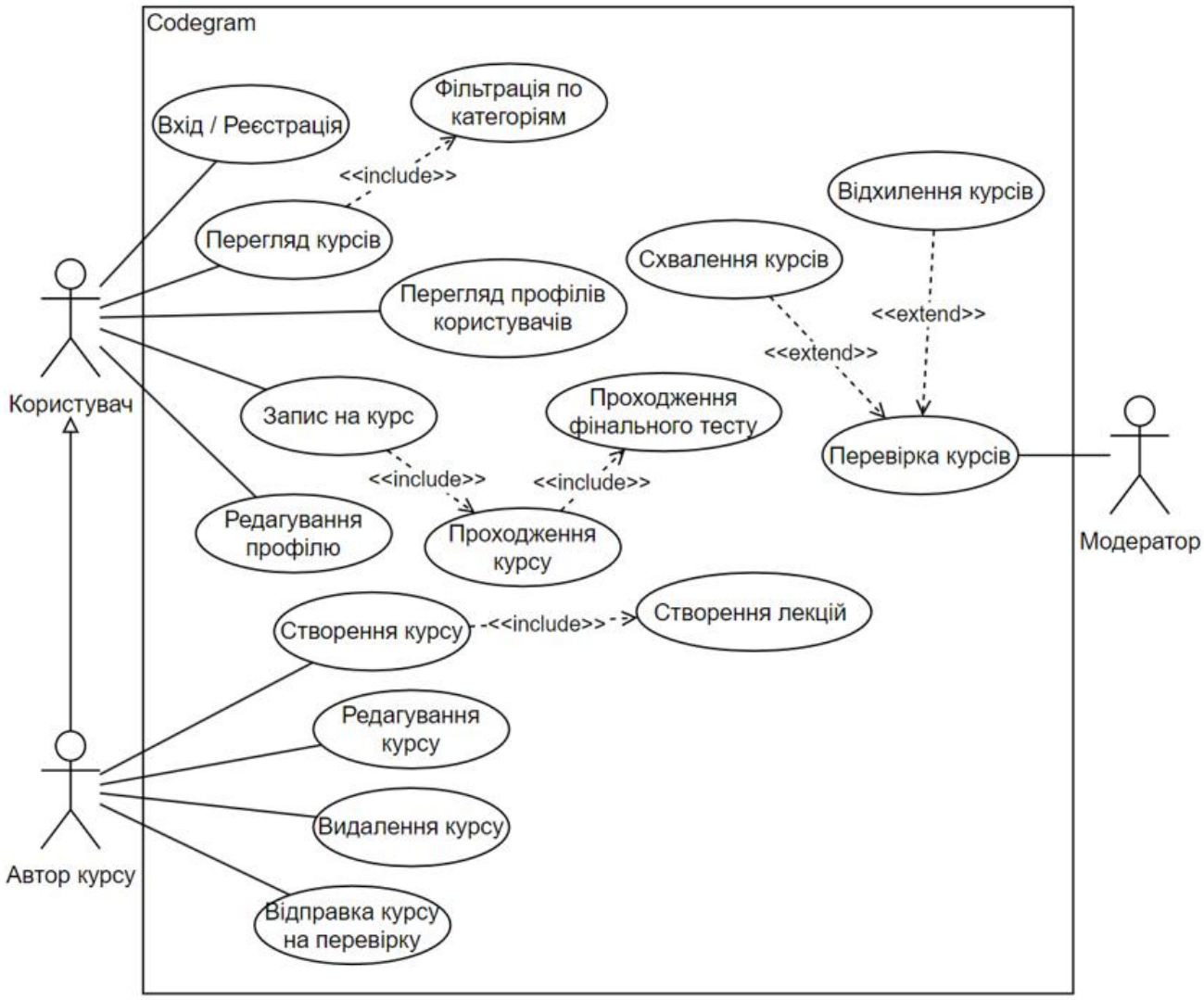


Рис. 2.1 Діаграма варіантів використання платформи Codegram

На даній діаграмі представлено три типи акторів: Користувач, Автор курсу та Модератор.

Варіанти використання для Користувача:

- Вхід/Реєстрація: Процес створення облікового запису або входу до системи для отримання доступу до функціональності платформи.
- Перегляд курсів: Дозволяє користувачеві переглядати доступні навчальні курси та включає в себе можливість фільтрації по категоріям.
- Запис на курс: Дозволяє користувачеві зареєструватися на обраний курс для подальшого навчання.

- Проходження курсу: Дозволяє користувачеві вивчати матеріали зареєстрованого курсу та включає в себе проходження фінального тесту після завершення навчання.
- Перегляд профілів користувачів: Дозволяє користувачеві переглядати профілі інших користувачів платформи.
- Редагування профілю: Дозволяє користувачеві оновлювати інформацію у власному профілі.

Варіанти використання для Автора курсу:

- Створення курсу: Дозволяє автору створювати новий навчальний курс і включає в себе створення лекцій для курсу.
- Редагування курсу: Дозволяє автору вносити зміни до створеного курсу.
- Видалення курсу: Дозволяє автору видаляти власні курси з платформи.
- Відправка курсу на перевірку: Дозволяє автору надсилати створений курс модератору для перевірки та затвердження.

Варіанти використання для Модератора:

- Перевірка курсів: Дозволяє модератору перевіряти якість та відповідність курсів стандартам платформи.
- Схвалення курсів: Дозволяє модератору затверджувати курси для публікації на платформі.
- Відхилення курсів: Дозволяє модератору відхиляти курси, які не відповідають стандартам платформи.

2.3 Проєктування архітектури web-платформи

Освітня web-платформа Codegram має клієнт-серверну архітектуру, що забезпечує оптимальний розподіл навантаження та легкість у розробці. Архітектура поділяється на дві основні частини, які взаємодіють між собою:

Клієнтська частина (Frontend): Розроблена з використанням JavaScript-фреймворку Vue.js, ця частина відповідає за відображення користувацького інтерфейсу у web-браузері та обробку взаємодії користувача із системою.

Використання Pinia забезпечує ефективне управління станом додатку, а Vite забезпечує швидку збірку та оптимізацію коду. Клієнтська частина надсилає запити до Firebase для отримання та збереження даних, а також відображає отриману інформацію користувачеві.

Серверна частина (Backend): Реалізована за допомогою Firebase як BaaS рішення, що надає повний набір інструментів для розробки серверної логіки без необхідності налаштування та управління власними серверами. Firebase забезпечує аутентифікацію користувачів, зберігання даних у Firestore (NoSQL база даних), зберігання файлів у Firebase Storage, а також надає API для взаємодії з клієнтською частиною. Обмін даними між клієнтом та Firebase відбувається через Firebase SDK у форматі JSON через захищене HTTPS з'єднання.

Ця архітектура забезпечує наступні переваги:

1. Розділення відповідальності: Клієнтська частина відповідає за відображення та взаємодію з користувачем, а Firebase – за зберігання та обробку даних, що покращує організацію коду та полегшує розробку.
2. Масштабованість: Firebase автоматично масштабується відповідно до навантаження, забезпечуючи стабільну роботу системи навіть при збільшенні кількості користувачів.
3. Швидкість розробки: Використання готової інфраструктури Firebase дозволяє зосередитися на розробці функціональності, а не на налаштуванні та підтримці серверів.
4. Синхронізація в реальному часі: Firestore забезпечує синхронізацію даних між всіма підключеними клієнтами в реальному часі, що є важливим для освітньої платформи.
5. Безпека: Firebase надає вбудовані механізми для автентифікації та авторизації користувачів, а також для захисту даних.

2.4 Проєктування структури бази даних

Проєктування структури бази даних є важливим етапом розробки освітньої web-платформи. Цей розділ описує деталі створення та організації бази даних, що забезпечує ефективне зберігання та обробку даних для підтримки функціональних вимог системи.

2.4.1 Вибір бази даних

Для проєкту обрано NoSQL базу даних Firestore. Вибір обумовлений низкою переваг, серед яких відсутність необхідності налаштування та управління власними серверами, можливість масштабування відповідно до зростання кількості користувачів, вбудована підтримка автентифікації користувачів, інтеграція з іншими сервісами Firebase (Authentication, Storage), можливість синхронізації даних у реальному часі та сумісність з Vue.js.

2.4.2 Структура бази даних

Проєктування структури бази даних включає створення різних колекцій, які зберігають різні типи даних, що необхідні для функціонування системи. На основі розроблених вимог було визначено наступні основні колекції:

Колекція User: Ця колекція зберігає інформацію про користувачів системи.

Поля:

- _id: string - унікальний ідентифікатор користувача;
- username: string - ім'я користувача;
- email: string - електронна пошта;
- role: string - роль користувача (user, moderator);
- authType: string - тип автентифікації (email, google);
- profile: object - додаткова інформація про користувача (bio, avatarUrl);
- createdAt: timestamp - дата створення облікового запису.

Колекція Course: Зберігає інформацію про навчальні курси.

Поля:

- _id: string - унікальний ідентифікатор курсу;
- authorId: string - ідентифікатор автора курсу;
- title: string - назва курсу;
- description: string - опис курсу;
- level: string - рівень складності (beginner, intermediate, advanced);
- duration: number - тривалість курсу у годинах;
- technologies: array - масив технологій, які вивчаються у курсі;
- language: string - мова курсу;
- availableFrom: string - період доступності курсу;
- students: string - кількість студентів, які проходять курс;
- topics: array - масив тем, які охоплює курс;
- image: string - URL зображення для курсу;
- status: string - статус курсу (approved, pending, rejected);
- reviewedBy: string - ідентифікатор модератора, який перевірив курс;
- createdAt: timestamp - дата створення;
- updatedAt: timestamp - дата останнього оновлення.

Колекція Module: Зберігає інформацію про модулі курсів.

Поля:

- _id: string - унікальний ідентифікатор модуля;
- courseId: string - ідентифікатор курсу;
- title: string - назва модуля;
- description: string - опис модуля;
- orderIndex: number - порядковий номер модуля в курсі.

Колекція Lesson: Зберігає інформацію про уроки в модулях.

Поля:

- _id: string - унікальний ідентифікатор уроку;
- moduleId: string - ідентифікатор модуля;

- title: string - назва уроку;
- content: string - HTML-вміст уроку;
- orderIndex: number - порядковий номер уроку в модулі;
- duration: number - тривалість уроку в хвилинах.

Колекція Enrollment: Зберігає інформацію про зарахування користувачів на курси.

Поля:

- _id: string - унікальний ідентифікатор зарахування;
- userId: string - ідентифікатор користувача;
- courseId: string - ідентифікатор курсу;
- progress: number - відсоток прогресу проходження курсу (0-100);
- completed: boolean - статус завершення курсу;
- enrolledAt: timestamp - дата зарахування на курс;
- lastModuleId: string - ідентифікатор останнього відвіданого модуля;
- lastLessonId: string - ідентифікатор останнього відвіданого уроку.

Колекція: FinalQuiz Зберігає інформацію про фінальні тести курсів.

Поля:

- _id: string - унікальний ідентифікатор тесту;
- courseId: string - ідентифікатор курсу;
- questions: array - масив об'єктів з питаннями та варіантами відповідей;
- passScore: number - прохідний бал для успішного завершення курсу (у відсотках).

Для демонстрації вигляду бази даних загалом, наведено діаграму на рисунку 2.2.

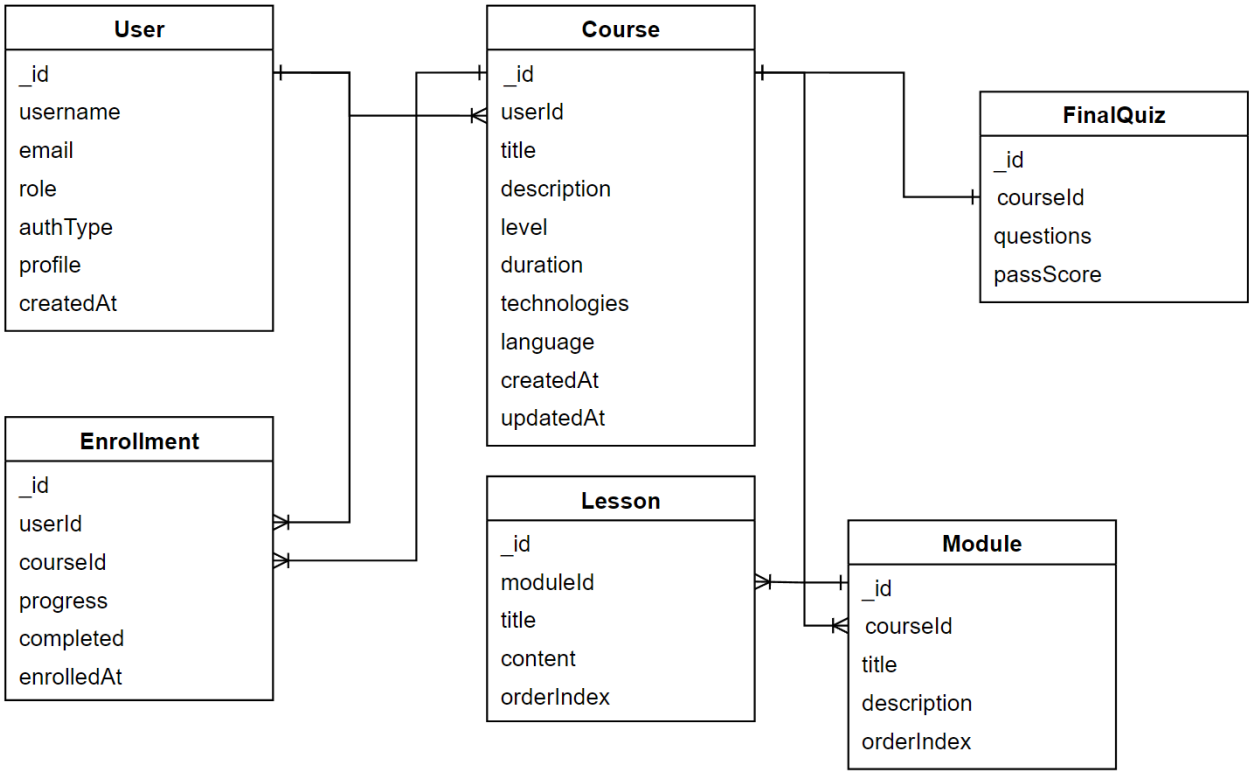


Рис. 2.2. Схема бази даних

Структура бази даних спроєктована з урахуванням зв'язків між сутностями, що забезпечує цілісність даних та ефективний доступ до інформації. Основні зв'язки між колекціями:

User → Course: один до багатьох (один користувач може створити багато курсів).

Course → Module: один до багатьох (один курс містить багато модулів).

Module → Lesson: один до багатьох (один модуль містить багато уроків).

Course → FinalQuiz: один до одного (один курс має один фінальний тест).

User → Enrollment ← Course: багато до багатьох (користувачі можуть бути зараховані на багато курсів, курс може мати багато зарахованих користувачів).

Така структура бази даних дозволяє ефективно реалізувати всі функціональні вимоги, забезпечуючи при цьому високу продуктивність та масштабованість системи.

2.5 Проєктування інтерфейсу користувача

Наступним кроком у розробці web-платформи була розробка концепції інтерфейсу користувача. На цьому етапі було розроблено зовнішній вигляд проєкту, способи взаємодії користувачів із системою та продумано основні сценарії використання платформи.

Для візуалізації дизайну було застосовано сучасний хмарний сервіс Figma, який надає розробникам інструментарій для проєктування та прототипування інтерфейсів. Ключовими перевагами Figma є можливість колективної роботи над проєктами в реальному часі та зберігання всіх матеріалів у хмарному середовищі, що значно спрощує процес розробки.

В процесі проєктування було створено основні екрани платформи (див. рисунок 2.3-2.4), які дозволили визначити оптимальне розташування функціональних блоків та елементів керування, а також забезпечити зручну навігацію між різними розділами платформи: головною сторінкою, каталогом курсів та сторінками авторизації/реєстрації.

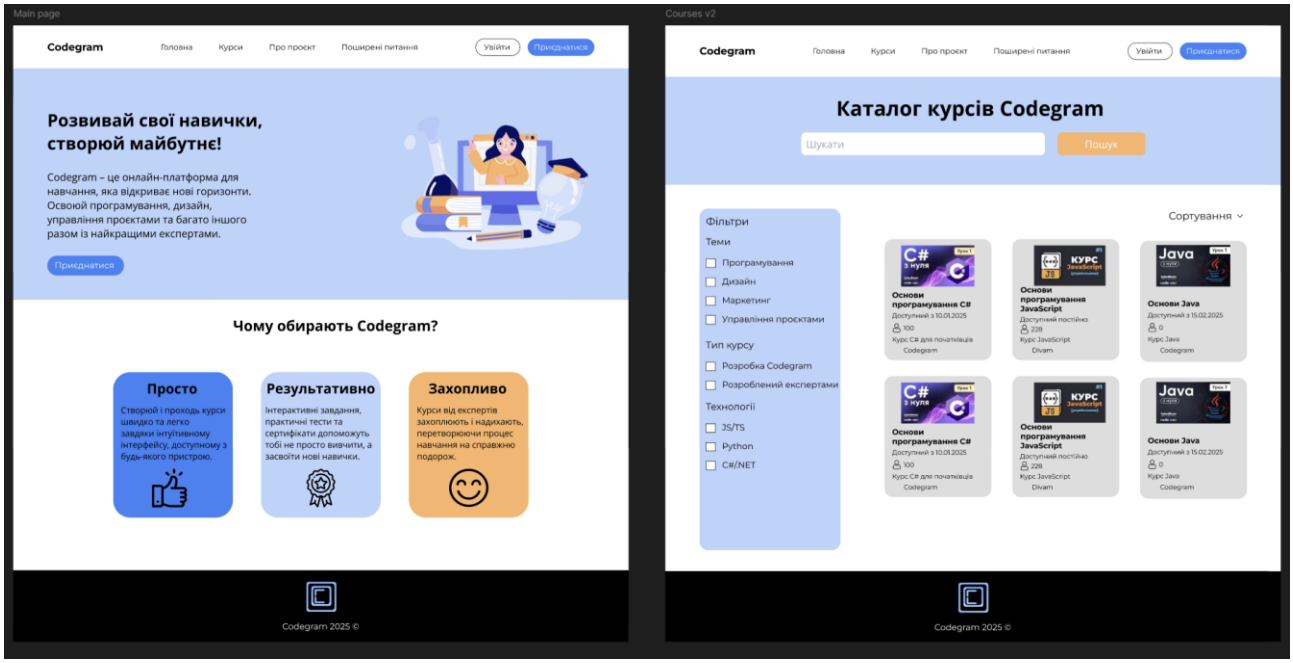


Рис. 2.3. Прототипи головної сторінки і каталогу курсів

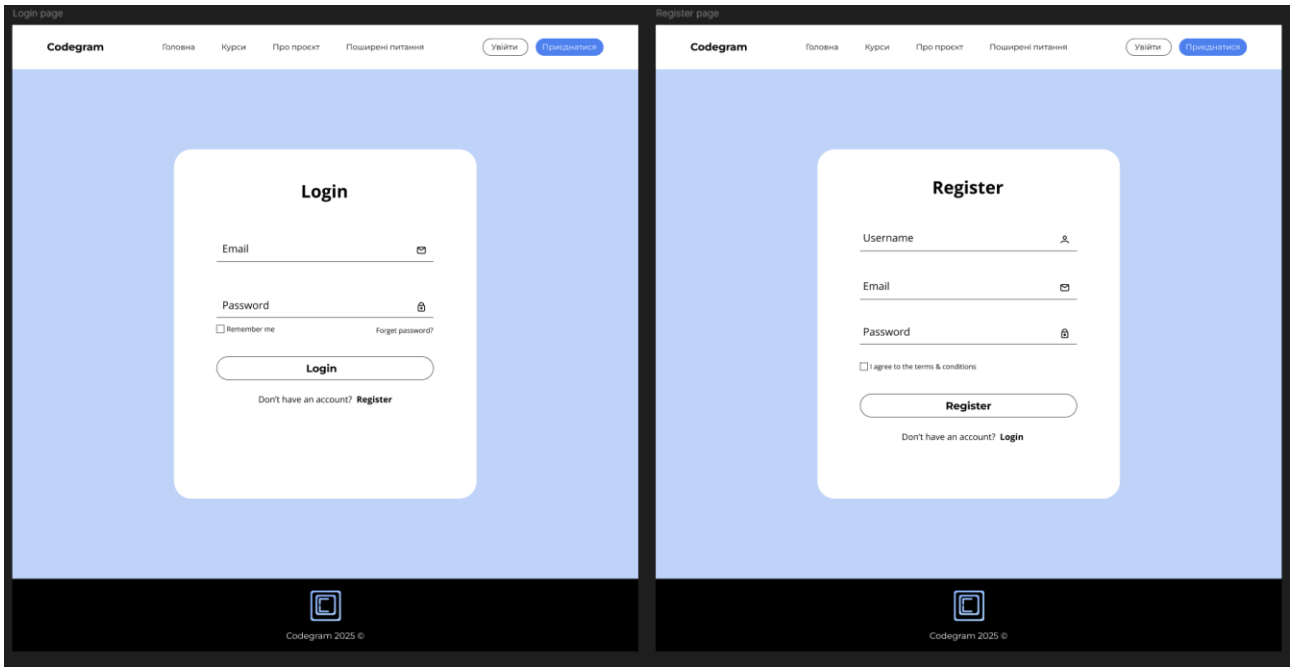


Рис. 2.4. Прототипи сторінок логіну та реєстрації

2.6 Мапа web-платформи

Мапа web-платформи Codegram представляє собою структурну схему організації сторінок та їх взаємозв'язків, що допомагає зрозуміти загальну архітектуру інтерфейсу та навігаційні шляхи користувачів. Мапа сайту є важливим інструментом для розробників, оскільки вона забезпечує розуміння організації контенту та функціональності платформи в залежності від стану автентифікації користувача.

2.6.1 Мапа сайту неавторизованого користувача

Структура сайту для неавторизованого користувача включає обмежений набір розділів, спрямований на ознайомлення нових відвідувачів з платформою та заохочення їх до реєстрації для отримання повного доступу до навчальних матеріалів (див. рисунок 2.5).

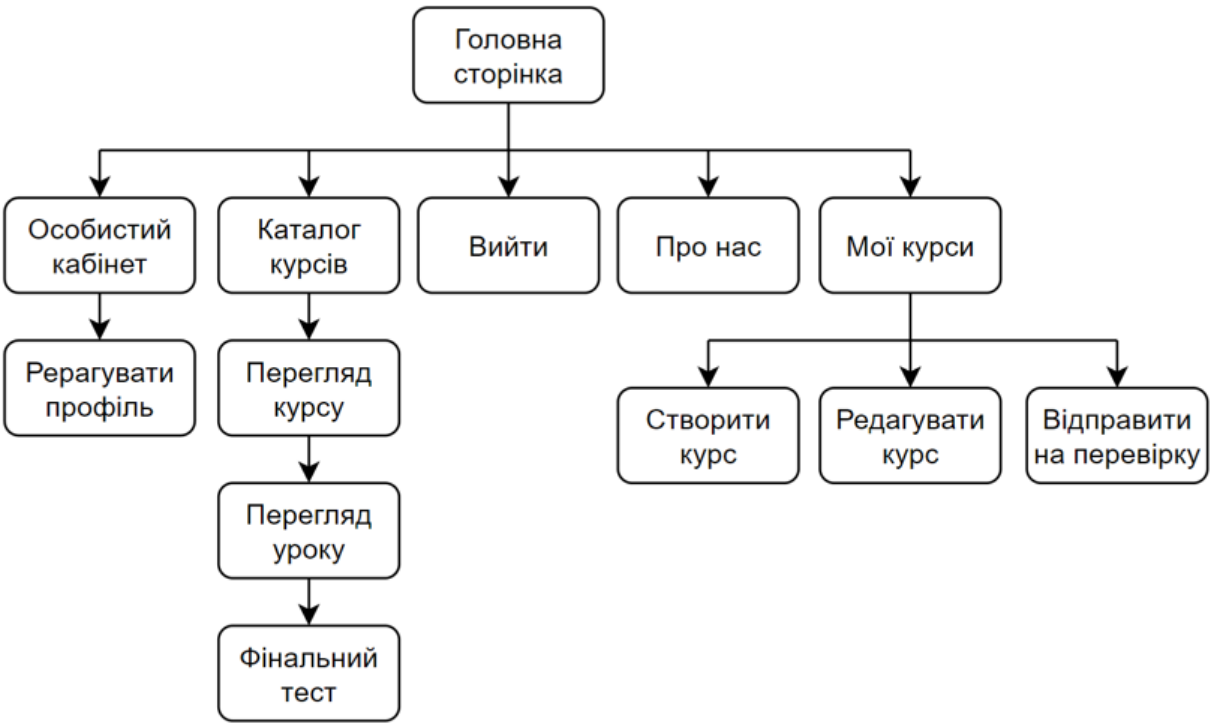


Рис. 2.5 Мапа для неавторизованого користувача

Мапа сайту для неавторизованого користувача включає наступні розділи:

- Головна сторінка: Містить загальну інформацію про платформу.
- Увійти: Сторінка авторизації для зареєстрованих користувачів.
- Зареєструватися: Форма реєстрації для нових користувачів.
- Про нас: Інформація про платформу Codegram.
- Каталог курсів: Перелік доступних курсів з можливістю перегляду основної інформації.
- Перегляд курсу: Сторінка з детальною інформацією про обраний курс.

2.6.2 Мапа сайту авторизованого користувача

Авторизовані користувачі отримують доступ до розширеної структури сайту, що включає повний функціонал платформи відповідно до ролі користувача (див. рисунок 2.6).

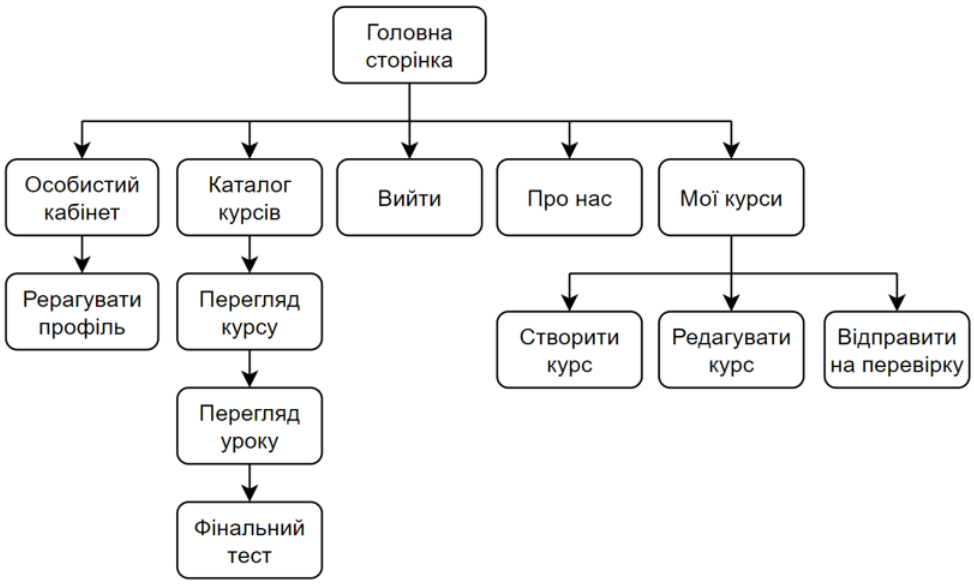


Рис. 2.6 Мапа для авторизованого користувача

Мапа сайту для авторизованого користувача включає наступні розділи:

- Головна сторінка: Містить загальну інформацію про платформу.
- Особистий кабінет: Доступ до особистої інформації та налаштувань.
- Редагувати профіль: Можливість оновлення особистих даних та налаштувань.
- Каталог курсів: Повний доступ до всіх навчальних курсів з можливістю фільтрації та пошуку.
- Перегляд курсу: Детальна інформація про курс із можливістю запису.
- Перегляд уроку: Доступ до навчальних матеріалів конкретного уроку.
- Фінальний тест: Завершальний тест для оцінки знань після проходження курсу.
- Вийти: Функція виходу з облікового запису.
- Про нас: Інформація про платформу.
- Мої курси: Розділ з курсами, створеними користувачем.
- Створити курс: Інтерфейс для створення нового навчального курсу.
- Редагувати курс: Можливість внесення змін до існуючих курсів.
- Відправити на перевірку: Функція для подання курсу на модерацію.

3 ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ WEB-ПЛАТФОРМИ

Розробка web-платформи для освітніх IT-курсів вимагає ретельного підбору технологій, які забезпечать повноцінну функціональність, високу продуктивність та комфортну взаємодію для користувачів. Важливо враховувати особливості освітніх систем, що працюють з різноманітним контентом і повинні гарантувати простоту у користуванні, ефективність навчального процесу, можливість масштабування та стабільну роботу.

3.1 Мова програмування JavaScript

JS є основною мовою програмування, що використовується для розробки клієнтської частини. Ця мова програмування була створена спеціально для розробки інтерактивних вебсайтів і сьогодні є однією з найпопулярніших мов програмування у світі.

JavaScript - це високорівнева, динамічна, об'єктно-орієнтована, прототипна мова програмування. Вона дозволяє створювати динамічні та інтерактивні web-інтерфейси, обробляти події, маніпулювати HTML-елементами та взаємодіяти з сервером за допомогою AJAX-запитів [11].

JS відрізняється винятковою гнучкістю у своїх можливостях, що відкриває простір для реалізації технік, недоступних у більш строго типізованих мовах, та сприяє створенню ефективного й адаптивного стилю програмування [12, с. 6]. Ця характеристика робить JavaScript ідеальним вибором для розробки web-платформи, де важлива швидка адаптація до змінних вимог та ефективна взаємодія з різними компонентами системи.

Універсальність JS забезпечує сумісність з різними пристроями та операційними системами, оскільки ця мова підтримується всіма сучасними браузерами. Широка екосистема бібліотек та фреймворків, значно полегшує та прискорює процес розробки. Важливою перевагою JavaScript є підтримка

асинхронного програмування, що дозволяє ефективно обробляти запити до сервера та інші операції, які вимагають очікування.

JavaScript відрізняється гнучкістю та не обмежує розробників жорсткими правилами, дозволяючи використовувати різні парадигми програмування: об'єктно-орієнтовану, функціональну, реактивну.

У проєкті JS використовується для реалізації інтерактивних елементів інтерфейсу, обробки користувацьких дій, валідації форм та асинхронної взаємодії з Firebase – BaaS-платформою, що забезпечує серверну функціональність.

Мова JS постійно розвивається, і в проєкті використано можливості сучасних стандартів ES6+, такі як стрілкові функції, деструктуризація об'єктів та масивів, шаблонні літерали, оператори розширення (spread/rest), асинхронні функції (async/await) та модульна структура для організації коду в окремі компоненти. Ці інновації забезпечують кращу підтримку для великих додатків, створення бібліотек та розширених можливостей програмування [13]. Використання сучасного JavaScript у поєднанні з інструментами збірки, такими як Vite, дозволяє створювати ефективний та оптимізований код.

3.2 Фреймворк Vue.js

Vue.js - це прогресивний JavaScript-фреймворк для створення користувацьких інтерфейсів [14]. На відміну від монолітних фреймворків, Vue спроєктований так, щоб його можна було поступово впроваджувати в проєкт. Його основна бібліотека зосереджена на шарі представлення, що спрощує інтеграцію з іншими бібліотеками та існуючими проєктами.

Для розробки платформи було обрано Vue.js 3, який надає низку суттєвих переваг. Компонентна архітектура Vue.js дозволяє розділити інтерфейс на незалежні компоненти, які можна використовувати повторно та тестувати окремо, що значно покращує організацію коду. Composition API Vue.js дозволяє логічно організувати код за функціональністю, а не за типом даних, що значно

спрощує розробку складних компонентів для освітньої платформи [15]. Vue.js демонструє високу продуктивність завдяки використанню віртуального DOM для оптимізації оновлень, що забезпечує швидку роботу навіть у складних додатках. Важливою перевагою є легкість вивчення та використання – Vue.js має інтуїтивно зрозумілий API та детальну документацію, що скорочує час на навчання нових розробників і дозволяє швидко розпочати роботу над проєктом.

В проєкті Vue.js використовується для створення інтерфейсу користувача, керування станом додатку та роутингу між різними сторінками. Для реалізації навігації між сторінками платформи було використано Vue Router [16], що забезпечує безшовний користувацький досвід без перезавантаження сторінки.

Разом з Vue.js використовується Vite — сучасний інструмент збірки для фронтенд-розробки, створений автором Vue.js [17]. Vite забезпечує миттєвий старт сервера розробки, використовуючи нативні ES-модулі браузера, що дозволяє уникнути повної збірки проєкту при запуску. HMR дозволяє миттєво відображати зміни в коді, а для продакшн-версії Vite створює оптимізований бандл з кодом, розділеним на чанки, що забезпечує швидке завантаження у браузері.

3.3 Бібліотека управління станом Pinia

Pinia — це бібліотека управління станом для Vue.js, яка прийшла на зміну Vuex у Vue 3 [18]. Вона була обрана для проєкту через свою простоту, типізацію та інтеграцію з Vue.js 3 Composition API.

Pinia пропонує низку переваг, які стали вирішальними при виборі цієї бібліотеки для проєкту. Простий та інтуїтивний API Pinia є більш оптимізованим порівняно з Vuex, що прискорює розробку та полегшує розуміння коду. Модульна структура Pinia дозволяє створювати незалежні сховища (stores), які можна тестувати окремо, що покращує підтримуваність коду. Повна інтеграція з Vue DevTools значно спрощує відлагодження додатку, дозволяючи стежити за

змінами стану в реальному часі. Порівняно невеликий розмір бібліотеки Pinia позитивно впливає на швидкість завантаження додатку в браузері.

У проєкті Pinia використовується для управління глобальним станом додатку, зокрема для збереження інформації про авторизованого користувача, управління кешуванням даних про курси, контролю стану завантаження та обробки помилок, а також зберігання прогресу користувача при проходженні курсів. Використання Pinia дозволило створити добре організовану архітектуру додатку, де компоненти можуть легко отримувати доступ до глобальних даних без зайвої передачі через ієрархію компонентів.

3.4 Хмарна платформа Firebase

Firebase - це платформа для розробки мобільних та web-додатків, що належить Google [19]. Вона надає набір інструментів та сервісів, які допомагають розробникам швидко створювати високоякісні додатки, зосереджуючись на розробці користувацького досвіду, а не на інфраструктурі.

Для проєкту було обрано Firebase як BaaS рішення завдяки суттєвим перевагам, які надає ця платформа. Firebase забезпечує високу швидкість розробки, дозволяючи уникнути написання власного бекенду з нуля та надаючи готові API для найбільш поширених задач. Важливою перевагою є вбудована масштабованість — Firebase автоматично адаптується до навантаження, що дозволяє обслуговувати як малу, так і велику кількість користувачів без додаткових налаштувань. Комплексність Firebase проявляється в об'єднанні різних сервісів, які зазвичай потрібні для web-додатків: база даних, автентифікація, хостинг, зберігання файлів — все це доступно в рамках єдиної платформи. Firebase надає інтегровані інструменти для безпеки даних та керування доступом, що спрощує захист користувацької інформації. Додатковими перевагами є вбудована аналітика та моніторинг, які дозволяють відстежувати продуктивність системи та аналізувати поведінку користувачів.

У проєкті використовуються різні компоненти Firebase. Firebase Authentication забезпечує реєстрацію та авторизацію користувачів, включаючи можливість входу через соціальні мережі. Firestore використовується як NoSQL база даних для зберігання даних про користувачів, курси, модулі, уроки та прогрес навчання. Firebase Storage надає можливість зберігання медіафайлів, пов'язаних із курсами, таких як зображення та PDF файли.

3.4.1 База даних Firestore

Firestore є базою даних платформи Firebase, яка використовується для зберігання всіх даних [20]. Це NoSQL база даних, що організовує дані у колекції та документи, забезпечуючи гнучку структуру даних.

Firestore має низку особливостей, які роблять її оптимальним вибором для проєкту освітньої платформи. Одна з ключових переваг — синхронізація в реальному часі, завдяки якій дані автоматично оновлюються між всіма підключеними клієнтами, що дозволяє створювати реактивні інтерфейси без додаткового програмування. Firestore підтримує гнучкі запити, сортування та фільтрацію, що дозволяє ефективно отримувати потрібні дані відповідно до різноманітних критеріїв. Система Firebase Security Rules забезпечує захист даних на рівні бази даних, контролюючи доступ до інформації на основі користувацьких ролей та інших умов [21]. Важливою характеристикою є автоматична масштабованість Firestore, яка не потребує додаткового налаштування серверів при зростанні обсягу даних чи кількості користувачів.

3.5 Середовище розробки VS Code

VS Code - це безкоштовний редактор коду з відкритим вихідним кодом, розроблений Microsoft. Для проєкту VS Code було обрано як основне середовище розробки через його функціональність, гнучкість та велику екосистему розширень [22].

VS Code має низку значних переваг, які зробили його оптимальним вибором для розробки web-платформи. Вбудована підтримка JavaScript та можливість розширення для роботи з Vue.js забезпечує зручне програмування з підсвічуванням синтаксису та автодоповненням, що підвищує продуктивність розробки. Інтеграція з Git спрощує роботу з версіями коду та колаборацію в команді, дозволяючи ефективно відстежувати зміни та вирішувати конфлікти безпосередньо в редакторі. Величезна бібліотека розширень VS Code забезпечує можливість налаштування середовища під конкретні потреби проєкту, додаючи специфічну функціональність без переходу до інших інструментів. Наявність вбудованого терміналу дозволяє запускати команди безпосередньо з редактора, що значно прискорює процес розробки, тестування та відлагодження. Інструменти відлагодження VS Code для JavaScript-коду допомагають швидко знаходити та виправляти помилки, аналізуючи виконання програми крок за кроком.

Для проєкту Codegram було встановлено спеціальні розширення VS Code, які покращили процес розробки. Розширення Vue забезпечує повноцінну підтримку Vue 3, покращуючи автодоповнення та типізацію. ESLint здійснює статичний аналіз коду та допомагає підтримувати узгоджений стиль програмування в усьому проєкті. Розширення Prettier забезпечує автоматичне форматування коду за заданими правилами, що дозволяє всій команді дотримуватися єдиного стилю. Інтеграція з Firebase через відповідне розширення надає можливість керувати сервісами Firebase безпосередньо з редактора, без необхідності переходити на вебконсоль. Розширення GitLens розширює можливості роботи з Git, дозволяючи детально аналізувати історію змін та ефективніше співпрацювати в команді.

Використання VS Code з вказаними розширеннями значно підвищило ефективність розробки, забезпечило уніфікований стиль коду та допомогло своєчасно виявляти потенційні помилки, що в кінцевому результаті істотно скоротило час розробки та підвищило стабільність роботи платформи.

4 РЕАЛІЗАЦІЯ WEB-ПЛАТФОРМИ ДЛЯ НАДАННЯ ДОСТУПУ ДО ОСВІТНІХ IT-КУРСІВ

4.1 Структура проєкту

Web-платформа розроблена з використанням Vue.js для клієнтської частини та Firebase для серверної частини. Для ефективної організації коду використовується компонентний підхід, що дозволяє підтримувати принцип повторного використання елементів інтерфейсу (див. рисунок 4.1).

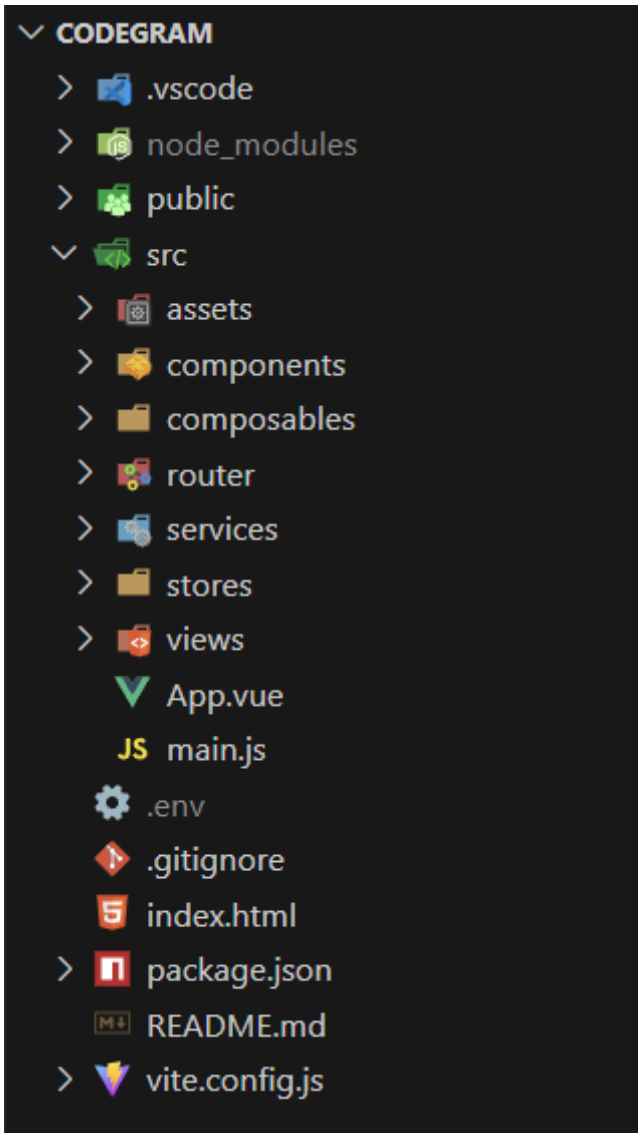


Рис. 4.1 Структура web-платформи Codegram

Директорія src є кореневою для вихідного коду проєкту та містить всі основні компоненти, сервіси та модулі системи. На кореневому рівні директорії src знаходяться файли App.vue - головний компонент застосунку, який містить основну структуру сторінки та main.js - точка входу в застосунок, де відбувається ініціалізація Vue.

Директорія assets містить зображення та SVG-іконки.

Директорія components містить набір повторно використовуваних компонентів інтерфейсу, які забезпечують модульність платформи (див. рисунок 4.2). Тут знаходяться компоненти навігації, елементи відображення курсів, фільтри пошуку, модальні вікна, індикатори завантаження та інші елементи інтерфейсу, які використовуються на різних сторінках системи. При розробці компонентів застосовувався підхід, за якого кожен елемент виконує чітко визначену функцію, що спрощує тестування, підтримку та розширення можливостей системи.

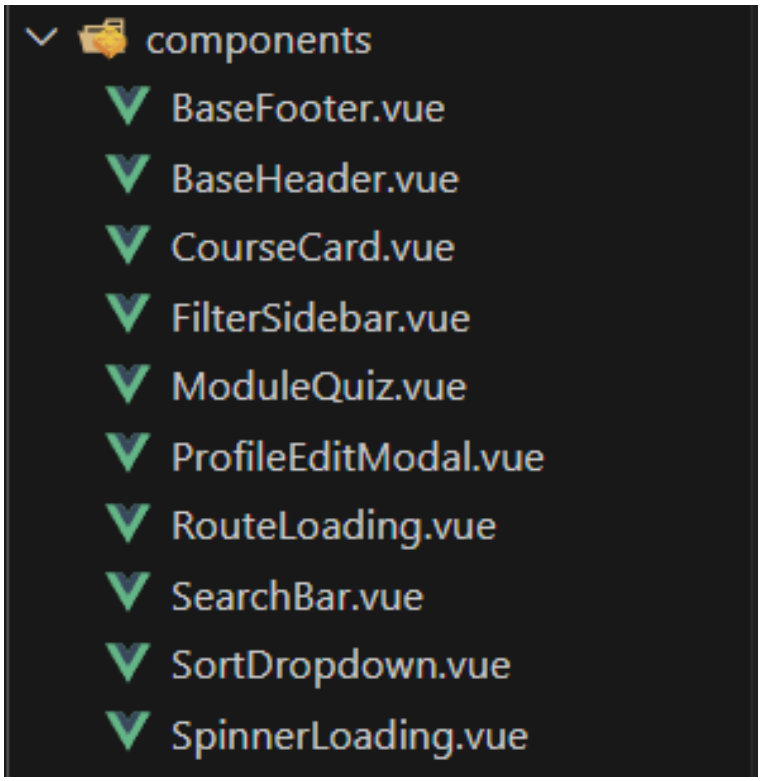


Рис. 4.2 Компоненти web-платформи

Директорія composables містить функції для роботи з даними, які можна використовувати на різних сторінках. Наприклад, файли useCourseDetail.js та useCourses.js містять логіку для роботи з курсами, яка використовується на різних сторінках. Це дозволяє не повторювати однаковий код на різних сторінках.

Директорія router відповідає за налаштування та управління маршрутизацією в застосунку. Вона містить конфігурацію всіх доступних маршрутів, обробку захисту маршрутів та перенаправлень в залежності від стану авторизації.

Система маршрутизації реалізує такі функції як маршрутизація між різними сторінками застосунку, захист маршрутів від неавторизованого доступу, обробка параметрів URL для динамічних сторінок, збереження історії навігації для можливості повернення та відображення прогресу завантаження при переході між сторінками.

Директорія services містить класи для роботи з базою даних Firebase. Ці класи спрощують взаємодію з базою даних для інших частин проєкту. Вони дозволяють компонентам отримувати та зберігати дані, не знаючи як саме відбувається робота з Firebase. Такий підхід також спрощує потенційну зміну провайдера бекенду без необхідності переписувати компоненти.

Директорія stores реалізує управління глобальним станом застосунку з використанням бібліотеки Pinia, де зберігаються стани для авторизації користувача, даних профілю користувача та інформації про записи на курси.

Використання глобального стану дозволяє різним компонентам отримувати доступ до спільних даних без необхідності передавати їх окремо(див. рисунок 4.3).

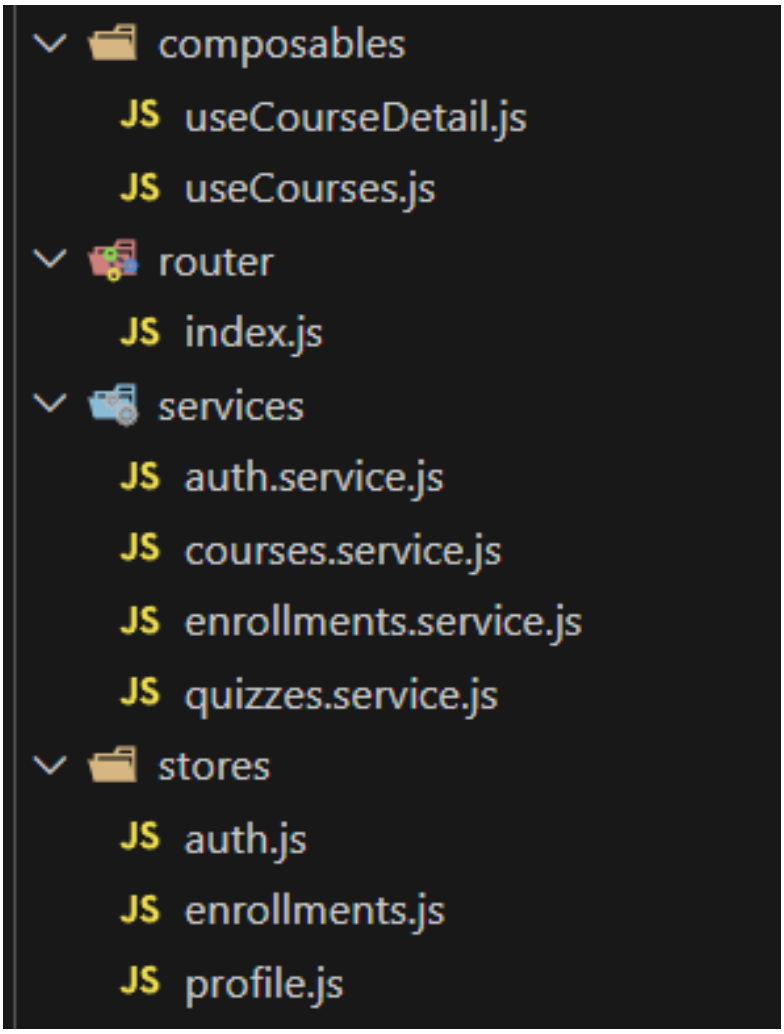


Рис. 4.3 Допоміжні модулі та сервіси web-платформи

Директорія views містить компоненти-сторінки застосунку. Сюди входять компоненти для головної сторінки, каталогу курсів з фільтрацією та сортуванням, детальної інформації про обраний курс, сторінки з матеріалами та завданнями уроку, сторінок авторизації, профілю користувача, управління створеними курсами, редактора для створення та модифікації курсів та адміністративної панелі для модераторів.

Ці компоненти-сторінки складніші та більші за розміром, ніж звичайні компоненти, оскільки вони координують роботу багатьох компонентів та управляють станом сторінки (див. рисунок 4.4).

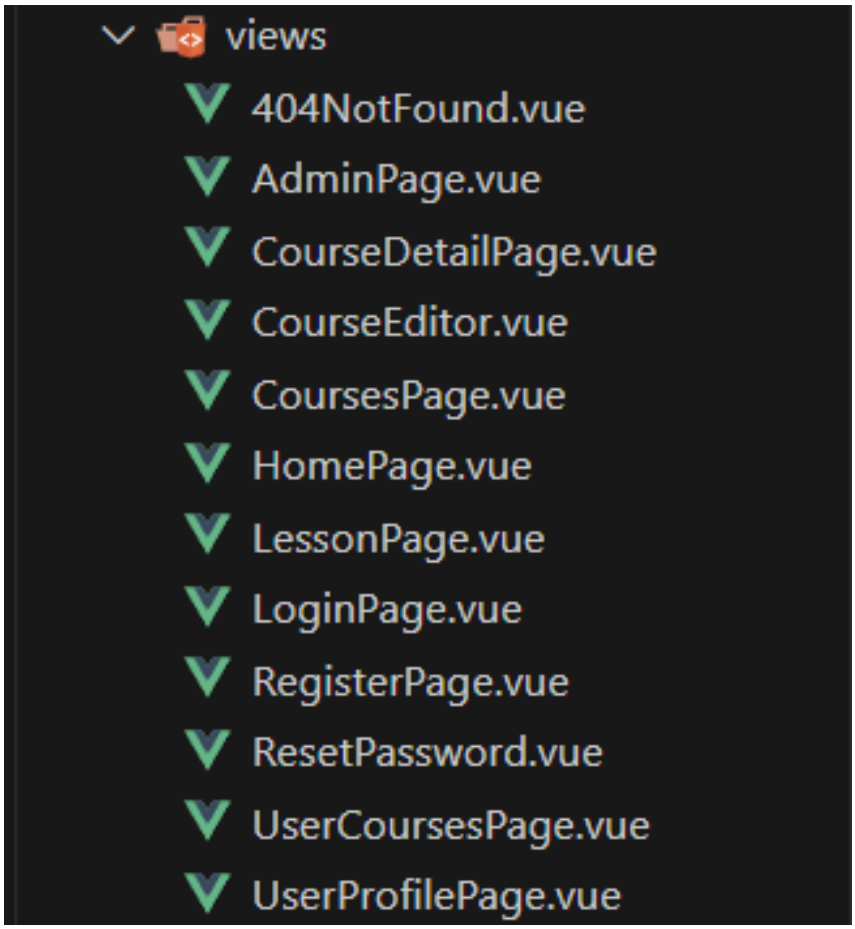


Рис. 4.4 Сторінки web-платформи

Така структура проєкту робить код зрозумілим і організованим. Вона допомагає легше створювати нові функції та виправляти помилки. Завдяки поділу на модулі, можна додавати нові можливості в систему без порушення роботи існуючих функцій.

4.2 Серверна частина

Реалізація серверної частини платформи Codegram базується на використанні Firebase - хмарної платформи від Google, що надає широкий спектр інструментів для розробки та хостингу web-застосунків. Використання Firebase дозволило зосередитися на розробці функціоналу замість налаштування та підтримки серверної інфраструктури.

4.2.1 Структура бази даних

Для зберігання даних платформи використовується Firestore - NoSQL база даних, що забезпечує гнучку структуру даних та можливість синхронізації в реальному часі (див. рисунок 4.5).

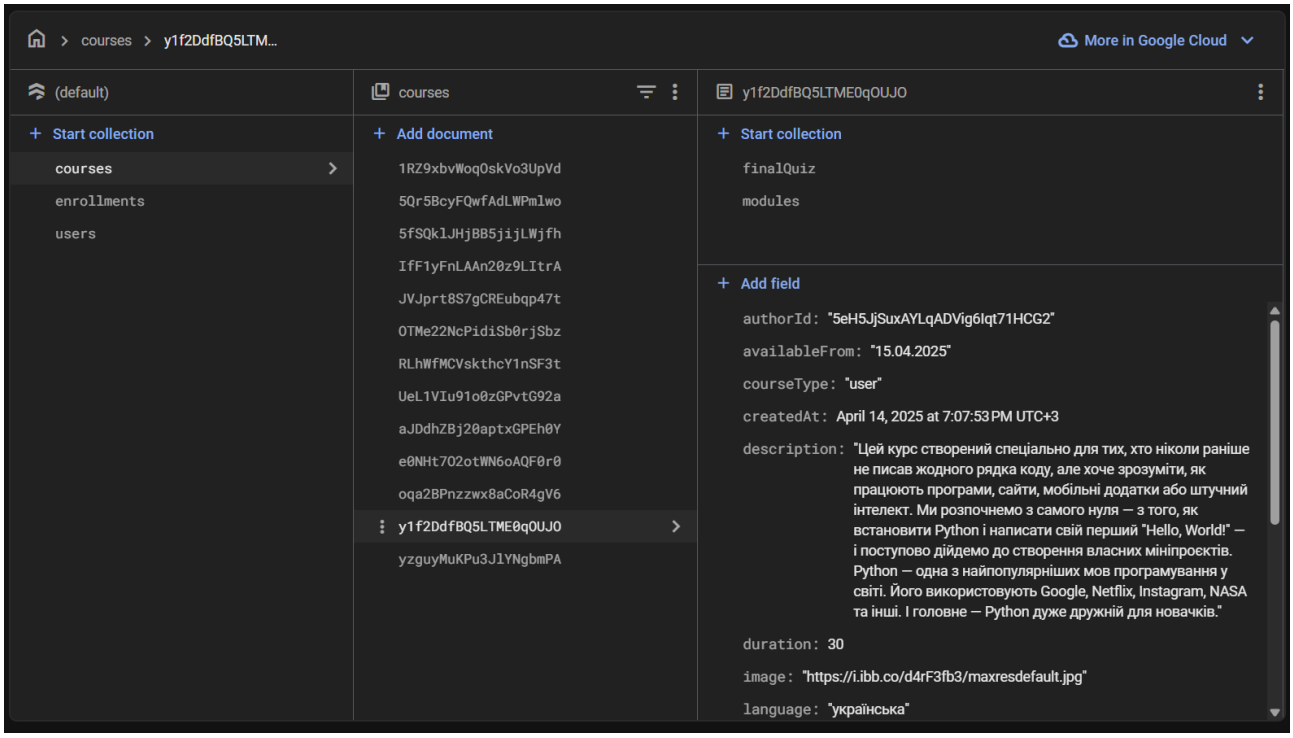


Рис. 4.5 Структура бази даних Firestore

Основні колекції бази даних включають кілька важливих структур. Колекція users зберігає інформацію про зареєстрованих учасників платформи: профілі, ролі, дати реєстрації та методи авторизації. Колекція courses містить детальну інформацію про кожен курс, включаючи назви, описи, зображення, рівні складності, мови, тривалості, тематики, технології та статуси.

Усередині курсів створюються підколекції modules, які представляють окремі модулі з їх описами та порядком проходження. В кожному модулі є підколекції lessons, які містять навчальні матеріали різних типів: текст, відео, PDF або комбіновані уроки.

Колекція enrollments відстежує записи користувачів на курси, фіксує прогрес навчання, останні переглянуті уроки та статуси завершення курсів.

Для кожного запису створюються підколекції completedLessons, що зберігають інформацію про виконані уроки та дати їх завершення. Підколекції quizResults зберігають результати проходження тестів з оцінками та відповідями користувачів.

Така структура бази даних забезпечує ефективне зберігання та швидкий доступ до всіх необхідних даних для функціонування освітньої платформи.

4.2.2 Система авторизації

Система авторизації є ключовим компонентом платформи, що забезпечує керування доступом та захист персональних даних користувачів. Для реалізації цієї системи використовується Firebase Authentication з додатковими власними механізмами контролю доступу (Рис. 4.6).

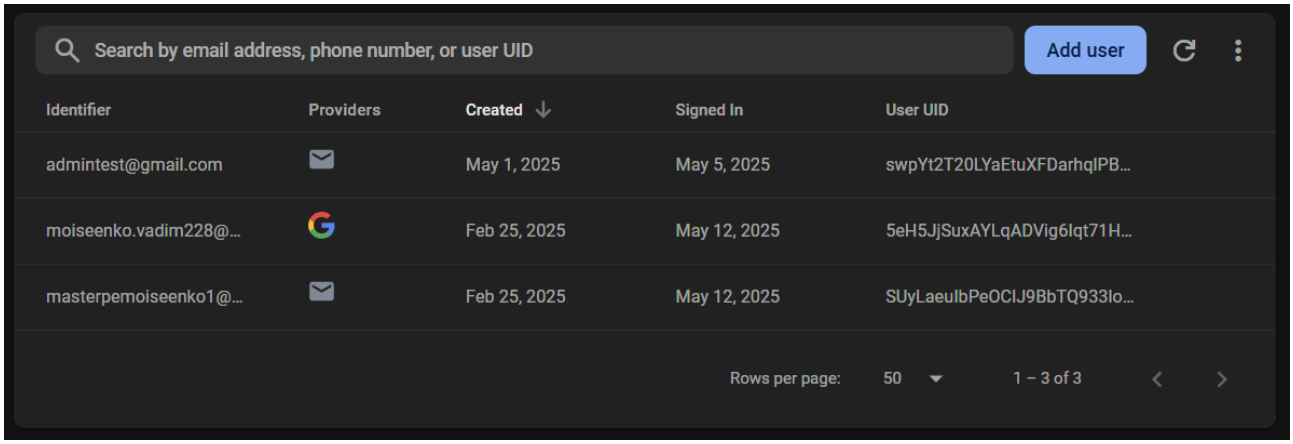


Рис. 4.6 Панель керування користувачами у Firebase Authentication

Система авторизації підтримує реєстрацію нових користувачів через електронну пошту з паролем або через Google-акаунт. При цьому здійснюється перевірка унікальності електронної пошти та логіну. Користувачі можуть проходити автентифікацію з різними методами входу, а також відновлювати доступ до облікового запису за допомогою електронної пошти.

Платформа реалізує систему ролей, яка визначає рівень доступу до функціональності. Користувач з роллю user має змогу проходити курси.

Роль moderator надає додаткові можливості для перевірки та схвалення курсів. Роль admin відкриває повний адміністративний доступ до всієї системи.

Безпека даних забезпечується як на клієнтській, так і на серверній стороні. Клієнтська частина використовує захист маршрутів через Vue Router, а серверна частина — правила безпеки Firestore, які визначають доступ до документів залежно від ролі користувача (див. рисунок 4.7).

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read: if true;
      allow create: if request.auth != null &&
                    request.auth.uid == userId;
      allow update, delete: if request.auth != null &&
                            request.auth.uid == userId;
    }

    match /courses/{courseId} {
      allow read: if true;
      allow create: if request.auth != null;
      allow update: if request.auth != null &&
                   (request.auth.uid == resource.data.authorId ||
                    request.auth.uid == request.resource.data.authorId ||
                    get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role in ['admin', 'moderator']);
      allow delete: if request.auth != null &&
                   (request.auth.uid == resource.data.authorId ||
                    get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role in ['admin', 'moderator']);

      match /modules/{moduleId} {
        allow read: if true;
        allow create: if request.auth != null;
        allow update, delete: if request.auth != null &&
                        (request.auth.uid == get(/databases/$(database)/documents/courses/$(courseId)).data.authorId ||
                         get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role in ['admin', 'moderator']);

        match /lessons/{lessonId} {
          allow read: if true;
          allow create: if request.auth != null;
          allow update, delete: if request.auth != null &&
                          (request.auth.uid == get(/databases/$(database)/documents/courses/$(courseId)).data.authorId ||
                           get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role in ['admin', 'moderator']);
        }
      }

      match /modules/{moduleId}/quiz {
        allow read: if true;
        allow create: if request.auth != null;
        allow update, delete: if request.auth != null &&
                        (request.auth.uid == get(/databases/$(database)/documents/courses/$(courseId)).data.authorId ||
                         get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role in ['admin', 'moderator']);
      }

```

Рис. 4.7 Правила безпеки Firestore

Такий дворівневий підхід гарантує, що користувачі мають доступ лише до тих даних і функцій, які відповідають їхнім повноваженням.

4.3 Клієнтська частина

Клієнтська частина платформи Codegram розроблена з використанням Vue.js 3 та організована за компонентним підходом, що забезпечує модульність та зручність підтримки коду.

Центральним компонентом застосунку є App.vue, який відповідає за ініціалізацію маршрутизації за допомогою Vue Router та визначення загальної структури сторінки. Цей компонент обробляє стан завантаження застосунку, відображає індикатори прогресу та забезпечує відображення відповідного контенту залежно від поточного маршруту (див. рисунок 4.8).

```vue
<template>
  <div id="app">
    <route-loading />
    <spinner-loading v-if="isLoading" />
    <router-view v-else></router-view>
  </div>
</template>

<script>
import { useAuthStore } from './stores/auth'
import SpinnerLoading from './components/SpinnerLoading.vue'
import RouteLoading from './components/RouteLoading.vue'

export default {
  name: 'App',
  components: {
    SpinnerLoading,
    RouteLoading,
  },
  Windsurf: Refactor | Explain | Generate Function Comment | ×
  data() {
    return {
      isLoading: true,
    }
  },
  Windsurf: Refactor | Explain | Generate Function Comment | ×
  async mounted() {
    const authStore = useAuthStore()
    try {
      await authStore.init()
      this.isLoading = false
    } catch (error) {
      console.error('Auth initialization error:', error)
      this.isLoading = false
    }
  },
}
</script>
```

Рис. 4.8 Маршрутизації у App.vue

Маршрутизація реалізована за допомогою Vue Router, який забезпечує навігацію між різними сторінками застосунку без перезавантаження сторінки. Маршрути визначені в файлі router/index.js, де також реалізовано захист маршрутів, перевірку автентифікації та обробку параметрів у URL (див. рисунок 4.9).

```
const routes = [
  { path: '/', component: HomePage },
  { path: '/courses', component: CoursesPage },
  { path: '/courses/:courseId', component: CourseDetailPage },
  {
    path: '/courses/:courseId/modules/:moduleId/lessons/:lessonId',
    component: LessonPage,
    meta: { requiresAuth: true },
    Windsurf: Refactor | Explain | Generate JSDoc | ×
    beforeEnter: async (to, from, next) => {
      const enrollmentsStore = useEnrollmentsStore()
      const authStore = useAuthStore()

      if (!authStore.isAuthenticated) {
        next('/login')
        return
      }

      const { isEnrolled } = await enrollmentsStore.checkEnrollment(to.params.courseId)
      if (!isEnrolled) {
        next(`/courses/${to.params.courseId}`)
        return
      }

      next()
    },
  },
  { path: '/login', component: LoginPage, meta: { requiresGuest: true } },
  { path: '/register', component: RegisterPage, meta: { requiresGuest: true } },
  { path: '/reset-password', component: ResetPassword, meta: { requiresGuest: true } },
  { path: '/profile', component: UserProfilePage, meta: { requiresAuth: true } },
  { path: '/user/:userId', component: UserProfilePage },
  { path: '/my-courses', component: UserCoursesPage, meta: { requiresAuth: true }, },
  { path: '/create-course', redirect: '/my-courses', meta: { requiresAuth: true },},
  { path: '/create-course/new', component: CourseEditor, meta: { requiresAuth: true },},
  { path: '/create-course/edit/:courseId', component: CourseEditor, meta: { requiresAuth: true },},
  { path: '/admin', component: AdminPage, meta: { requiresAuth: true, requiresAdmin: true },},
  { path: '/:pathMatch(.*)*', name: 'NotFound', component: () => import('../views/404NotFound.vue'),},
]
```

Рис. 4.9 Код маршрутизації в router/index.js

Система автентифікації забезпечує реєстрацію нових користувачів та вхід існуючих користувачів до платформи. Для цього реалізовано сторінки реєстрації, входу та відновлення паролю.

На сторінці реєстрації користувач може створити новий обліковий запис, вказавши логін, електронну пошту та пароль. Також реалізовано можливість реєстрації через Google аккаунт (див. рисунок 4.10).

Процес реєстрації забезпечується сервісом AuthHelper. При створенні нового облікового запису система перевіряє унікальність електронної пошти та логіну, а також створює відповідний документ у колекції users в базі даних.

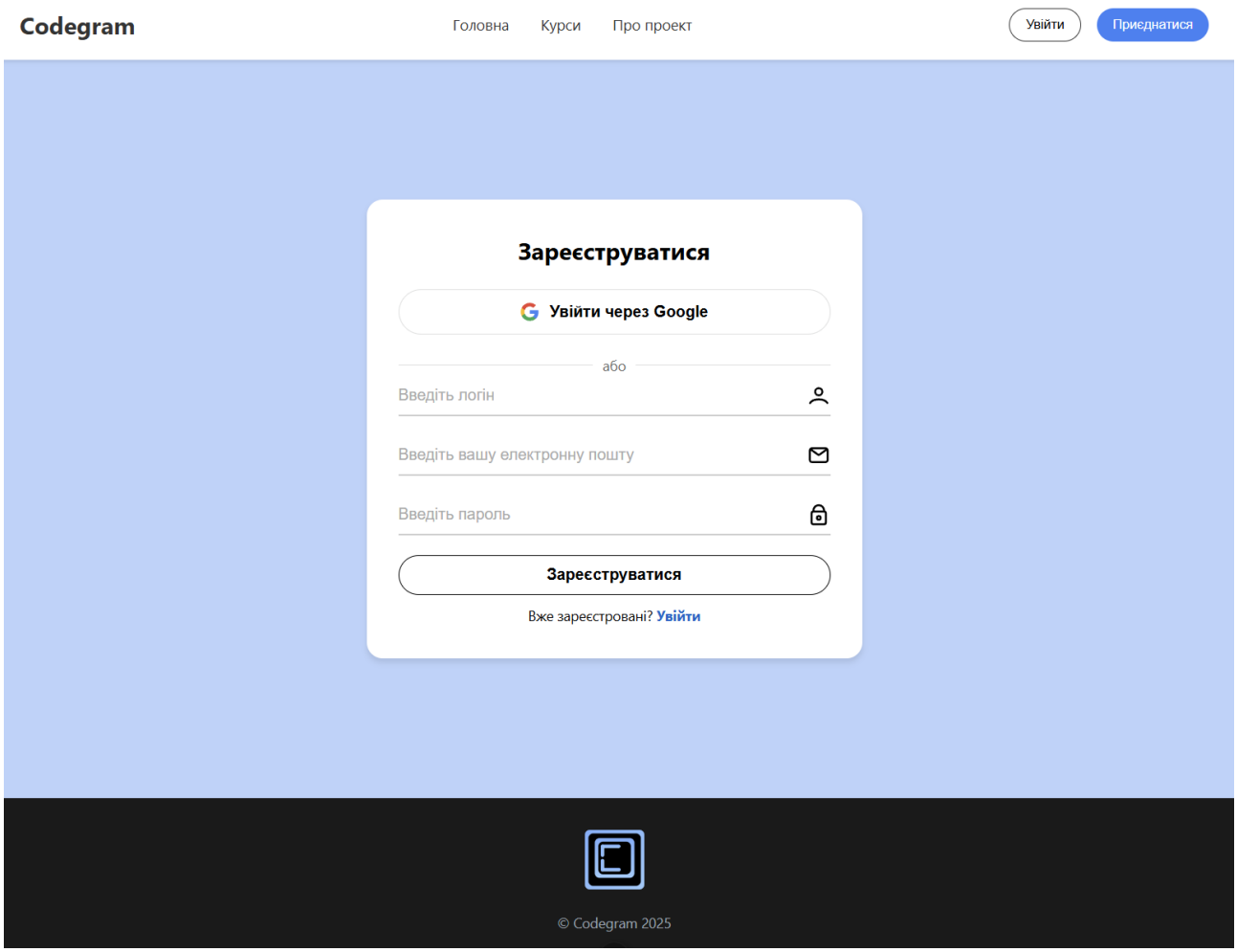


Рис. 4.10 Сторінка реєстрації користувача

Для користувачів, які вже мають обліковий запис, доступна сторінка входу. Вона містить форму для введення електронної пошти та паролю, а також кнопку для входу через Google. Для підвищення безпеки облікових записів реалізовано функціональність відновлення паролю. Користувачі, які забули свій пароль, можуть відправити запит на його відновлення через електронну пошту.

Каталог курсів дозволяє користувачам переглядати доступні курси з можливістю фільтрації та пошуку (див. рисунок 4.11).

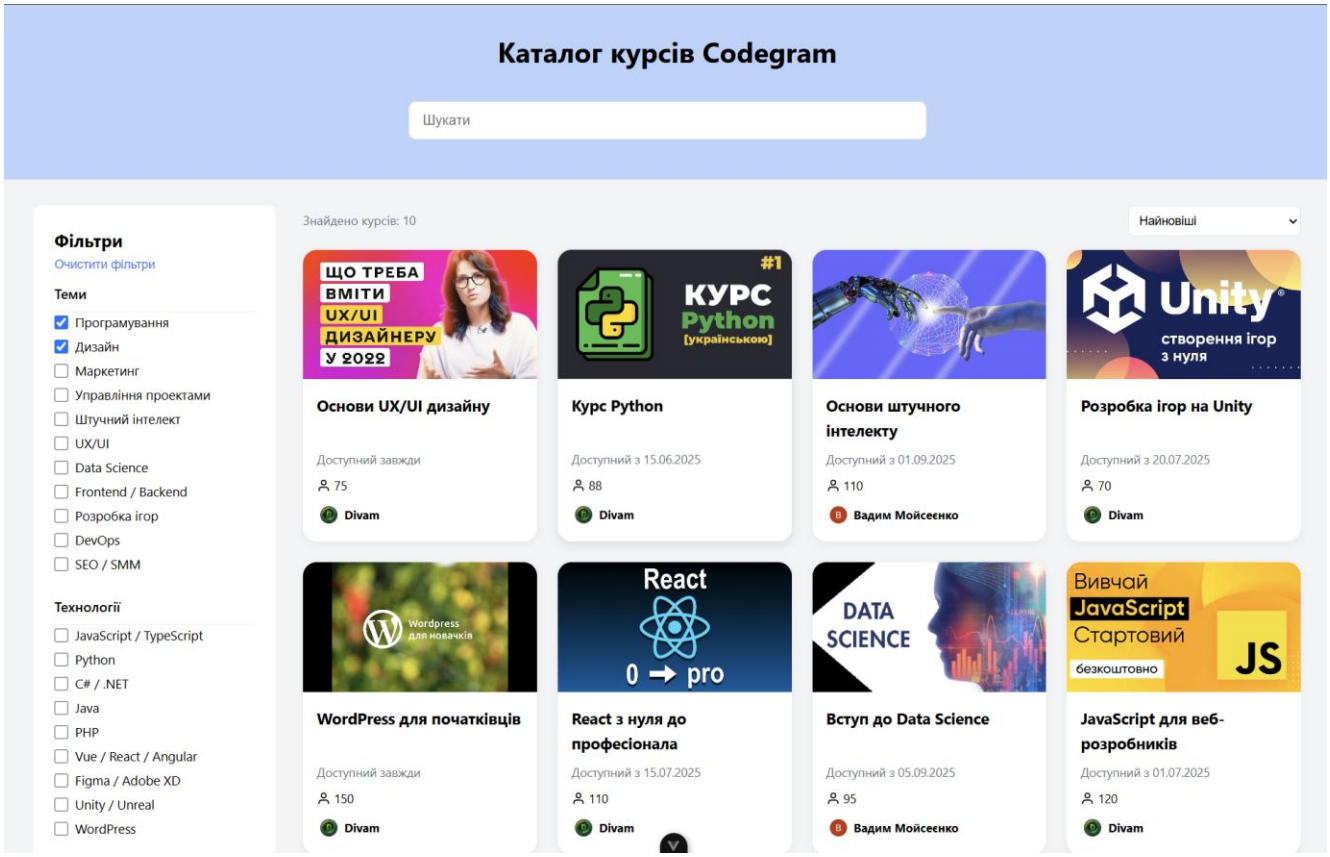


Рис. 4.11 Сторінка з Каталогом курсів

Реалізовано компонент FilterSidebar для фільтрації за різними параметрами: тематикою, технологіями, рівнем складності, мовою та тривалістю. Пошук курсів реалізовано за допомогою компонента SearchBar, який дозволяє шукати курси за назвою або ключовими словами. Результати пошуку оновлюються динамічно при введенні запиту.

При виборі конкретного курсу користувач потрапляє на сторінку з детальною інформацією. Компонент CourseDetailPage відображає повний опис курсу, включаючи зображення, автора, рівень складності, тривалість, мову та структуру (див. рисунок 4.12).

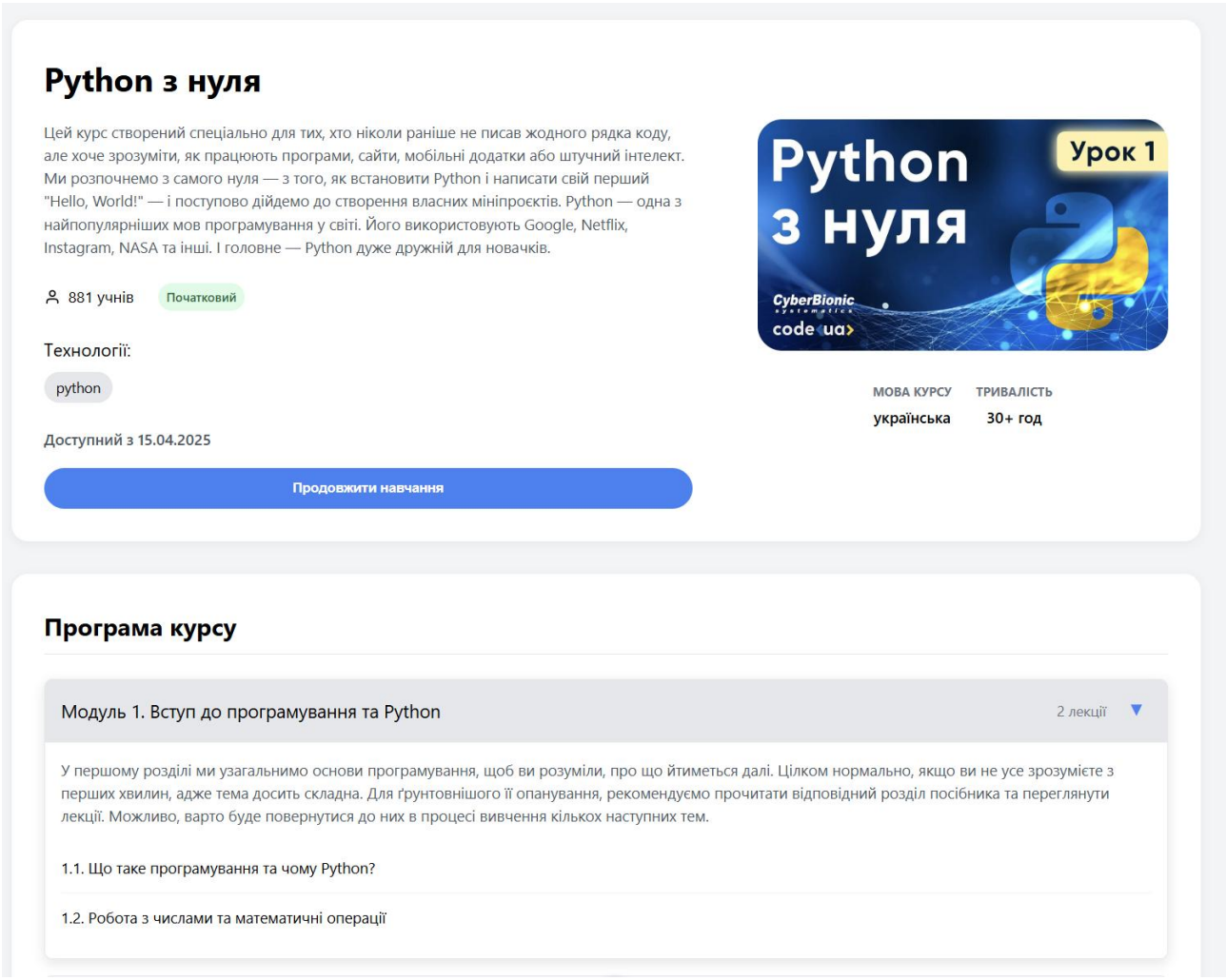


Рис. 4.12 Сторінка перегляду курсу

На сторінці курсу відображається список модулів та уроків, а також кнопка для запису на курс. Користувач може переглянути структуру курсу перед тим, як зареєструватися.

Функціональність запису на курс реалізована через сервіс EnrollmentsService, який створює запис у базі даних та зв'язує користувача з обраним курсом (див. рисунок 4.13).

```
75  async enrollCourse(userId, courseId) {
76    try {
77      const { isEnrolled, enrollment } = await this.checkEnrollment(userId, courseId)
78
79      if (isEnrolled) {
80        return enrollment
81      }
82
83      const enrollmentData = {
84        userId,
85        courseId,
86        enrolledAt: serverTimestamp(),
87        progress: 0,
88        completed: false,
89      }
90
91      const enrollmentsRef = collection(this.db, 'enrollments')
92      const docRef = await addDoc(enrollmentsRef, enrollmentData)
93
94      const courseRef = doc(this.db, 'courses', courseId)
95      await runTransaction(this.db, async (transaction) => {
96        const courseDoc = await transaction.get(courseRef)
97        if (!courseDoc.exists()) {
98          throw "Course doesn't exist!"
99        }
100
101       const courseData = courseDoc.data()
102       const currentStudents = courseData.students || 0
103
104       transaction.update(courseRef, {
105         students: currentStudents + 1,
106       })
107     })
108
109     return {
110       id: docRef.id,
111       ...enrollmentData,
112     }
113   } catch (error) {
114     console.error('Помилка при записі на курс:', error)
115     throw new Error(`Помилка при записі на курс: ${error.message}`)
116   }
117 }
```

Рис. 4.13 Реалізація функції, що реєструє користувача на курс

Після запису на курс користувач отримує доступ до навчальних матеріалів. Сторінка уроку (LessonPage) відображає контент уроку, який може бути у форматі тексту, відео або PDF-документа (див. рисунок 4.14).

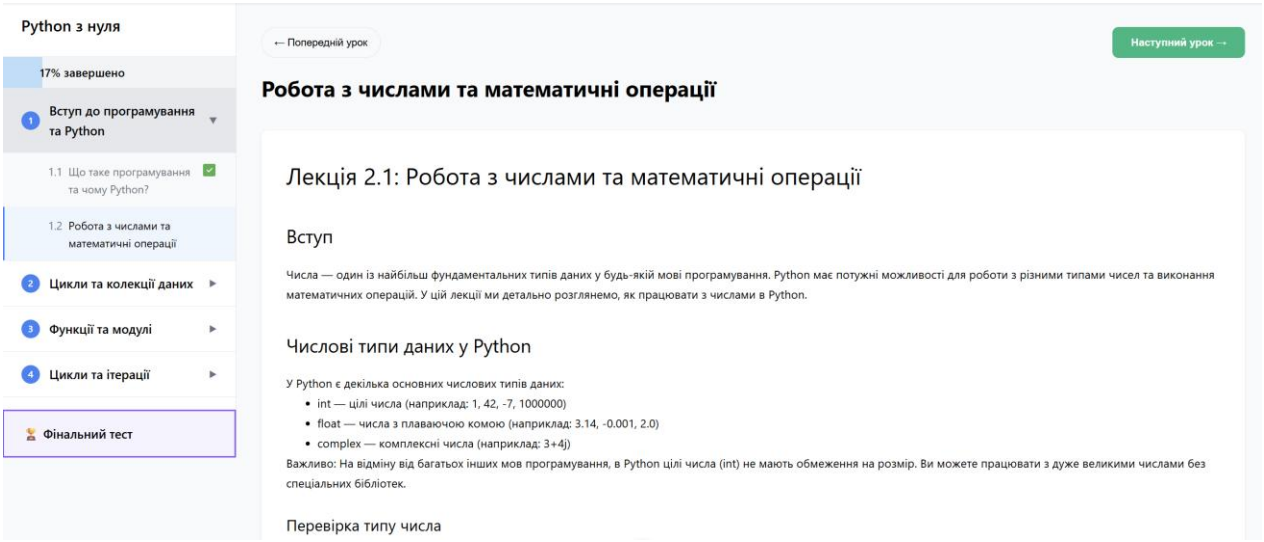


Рис. 4.14 Сторінка перегляду уроку

Для зручної навігації між уроками реалізовано бічну панель з структурою курсу. Користувачі можуть розгортати модулі та переходити між уроками, а також бачити свій прогрес проходження курсу.

Після завершення уроку користувач може позначити його як виконаний, що оновлює загальний прогрес курсу. Функціональність відстеження прогресу реалізована через сервіс EnrollmentsService, який зберігає інформацію про пройдені уроки та розраховує відсоток виконання курсу(див. рисунок 4.15).

```
158  async completeLesson(enrollmentId, moduleId, lessonId) {
159    try {
160      const completedLessons = await this.getCompletedLessons(enrollmentId)
161      const isCompleted = completedLessons.some(
162        (lesson) => lesson.id === lessonId && lesson.moduleId === moduleId,
163      )
164
165      if (isCompleted) {
166        return completedLessons.find(
167          (lesson) => lesson.id === lessonId && lesson.moduleId === moduleId,
168        )
169      }
170
171      const lessonData = {
172        moduleId,
173        completedAt: serverTimestamp(),
174      }
175
176      const lessonsRef = collection(this.db, `enrollments/${enrollmentId}/completedLessons`)
177      const lessonRef = doc(lessonsRef, lessonId)
178
179      await setDoc(lessonRef, lessonData, { merge: true })
180
181      await this.updateProgress(enrollmentId)
182
183      return {
184        id: lessonId,
185        ...lessonData,
186      }
187    } catch (error) {
188      console.error('Помилка при позначенні уроку як виконаного:', error)
189      throw new Error(`Помилка при позначенні уроку як виконаного: ${error.message}`)
190    }
191  }
```

Рис. 4.15 Реалізація функції для позначення уроку як виконаного

Для перевірки засвоєних знань після завершення курсу користувачу пропонується пройти фінальний тест. Компонент ModuleQuiz відображає питання з варіантами відповідей та підраховує результати (див. рисунок 4.16).

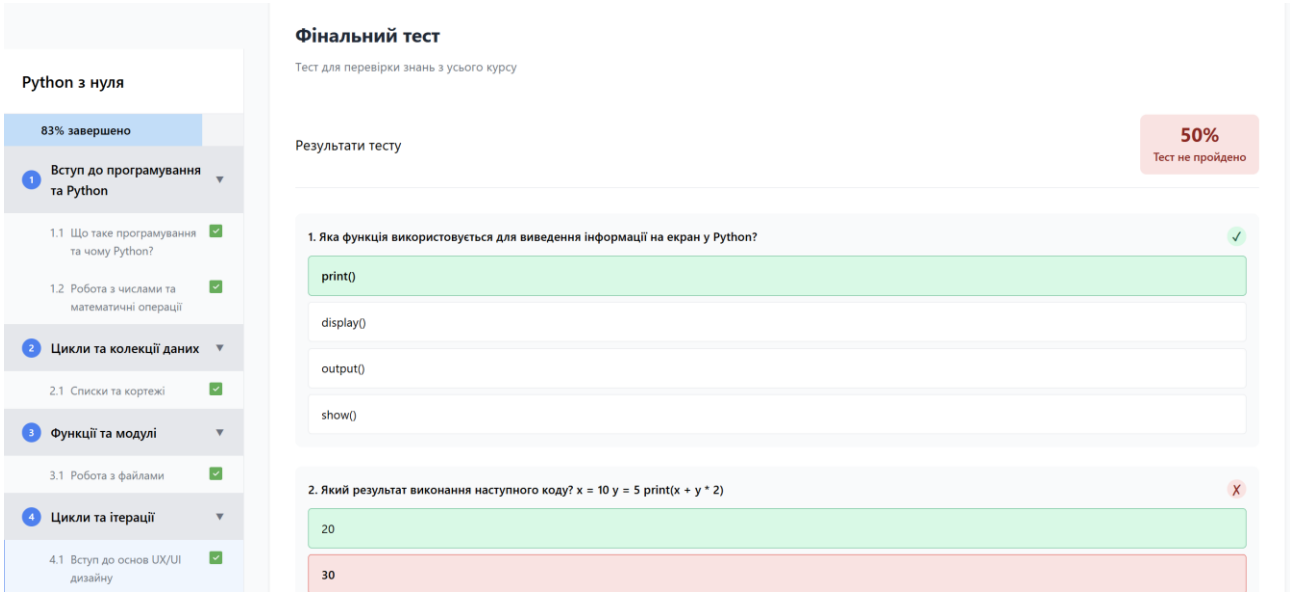


Рис. 4.16 Інтерфейс проходження фінального тесту

Після завершення тесту користувач отримує результати з відображенням правильних та неправильних відповідей. Якщо користувач набрав достатню кількість балів, курс позначається як завершений.

Особистий профіль користувача дозволяє керувати персональною інформацією та відстежувати прогрес навчання. Сторінка профілю відображає дані користувача, список курсів, на які він записаний, та курси, які він створив (див. рисунок 4.17).

Користувач може редагувати свою інформацію через модальне вікно редагування профілю. Компонент ProfileEditModal дозволяє змінювати ім'я, аватар та інформацію про себе (див. рисунок 4.18).

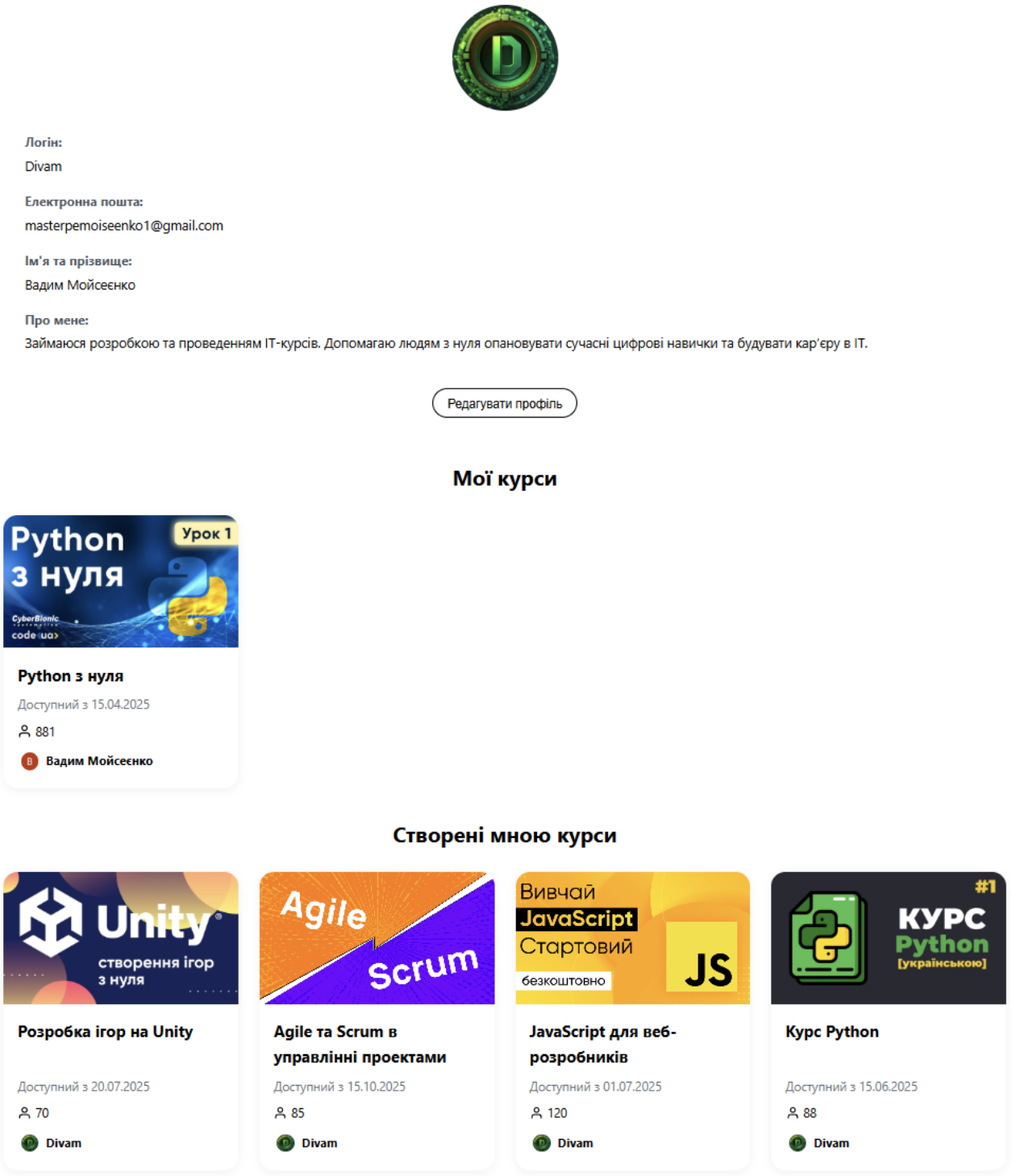


Рис. 4.17 Особистий кабінет користувача

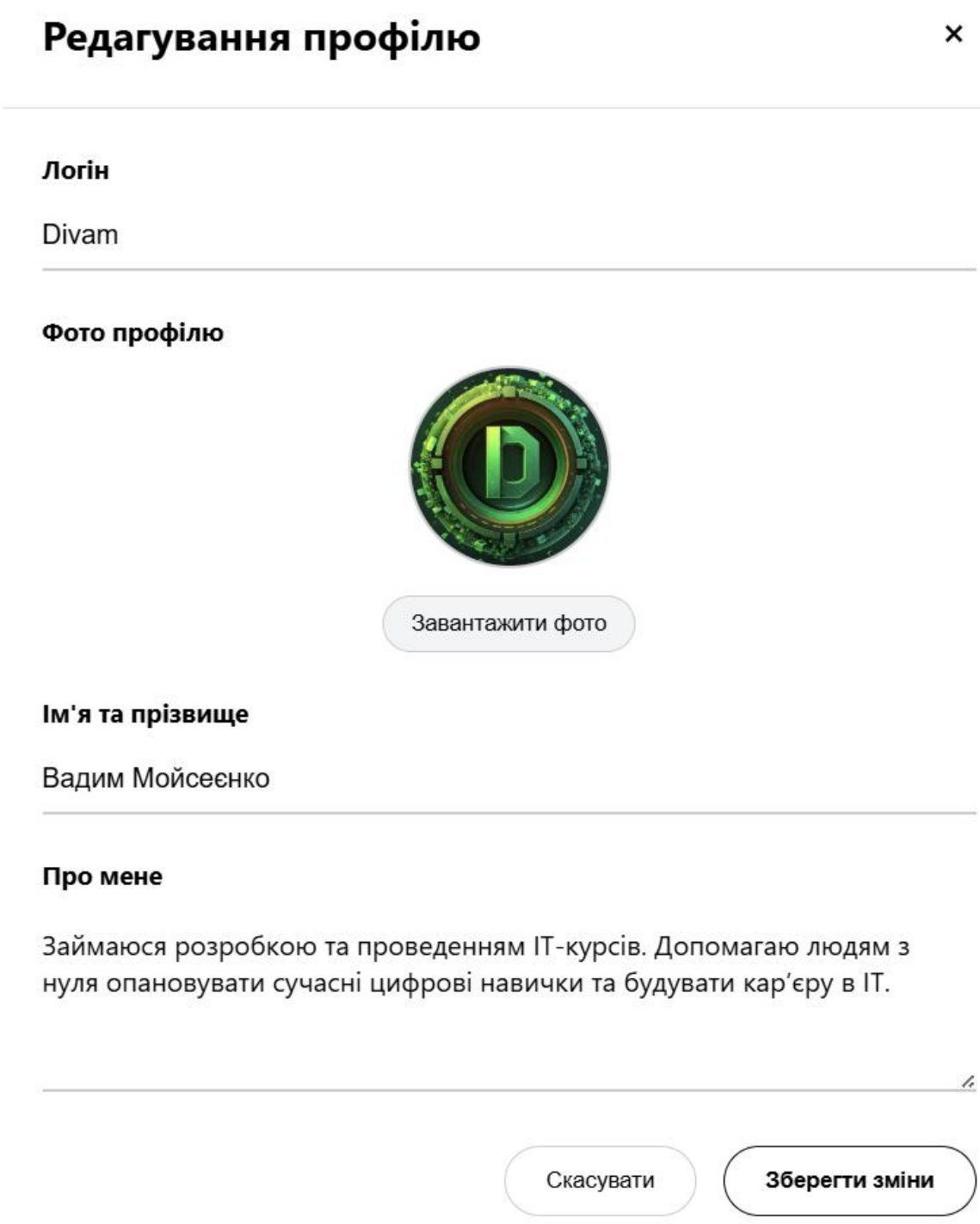


Рис. 4.18 Вікно редагування особистого профілю

Користувачі можуть створювати власні курси. Сторінка створення курсу (CourseEditor) надає інтерфейс для введення всіх необхідних даних: назви, опису, зображення, рівня складності, мови, тривалості, а також для додавання модулів та уроків (див. рисунок 4.19).

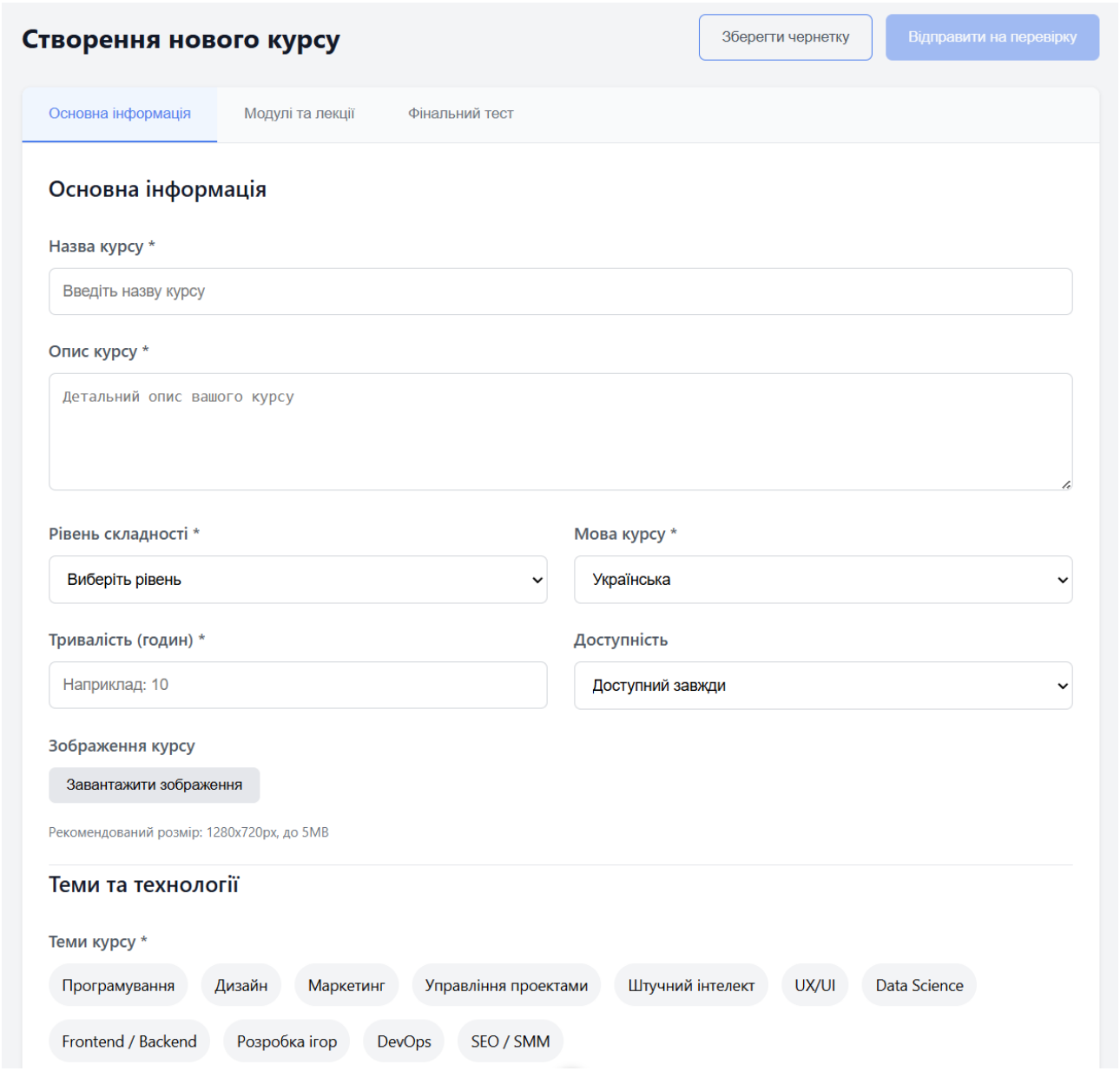


Рис. 4.19 Інтерфейс створення нового курсу

Редагування курсу здійснюється через той самий інтерфейс. Користувач може змінювати інформацію про курс, додавати або видаляти модулі та уроки.

Після створення або редагування курсу він відправляється на перевірку модератору. Статус курсу відображається в особистому кабінеті автора (див. рисунок 4.20).

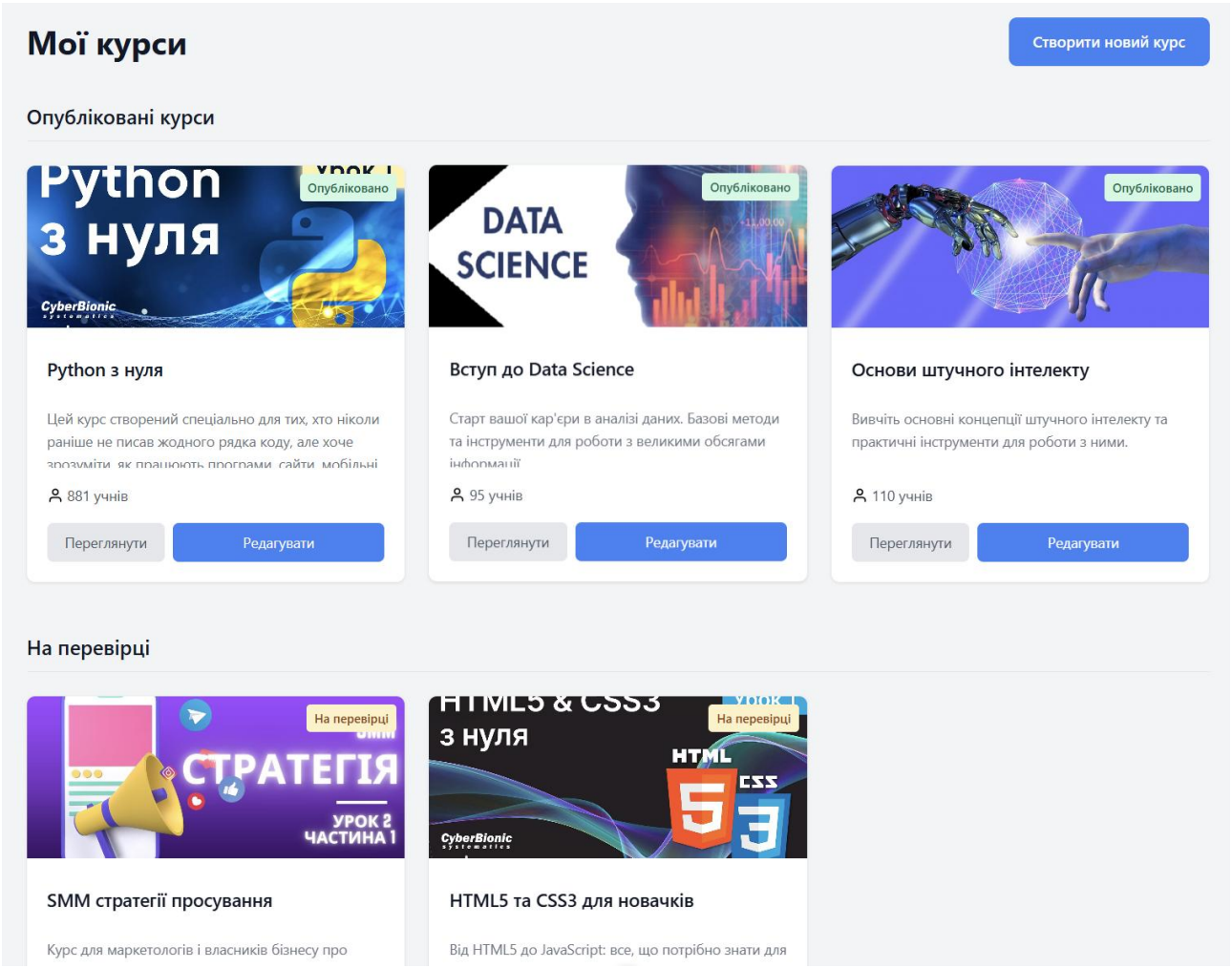


Рис. 4.20 Відображення створених курсів з різними статусами

Для модераторів реалізовано адміністративну панель, яка дозволяє перевіряти нові курси перед їх публікацією. Модератор може переглянути інформацію про курс, його структуру та контент, а потім схвалити або відхилити публікацію (див. рисунок 4.21).

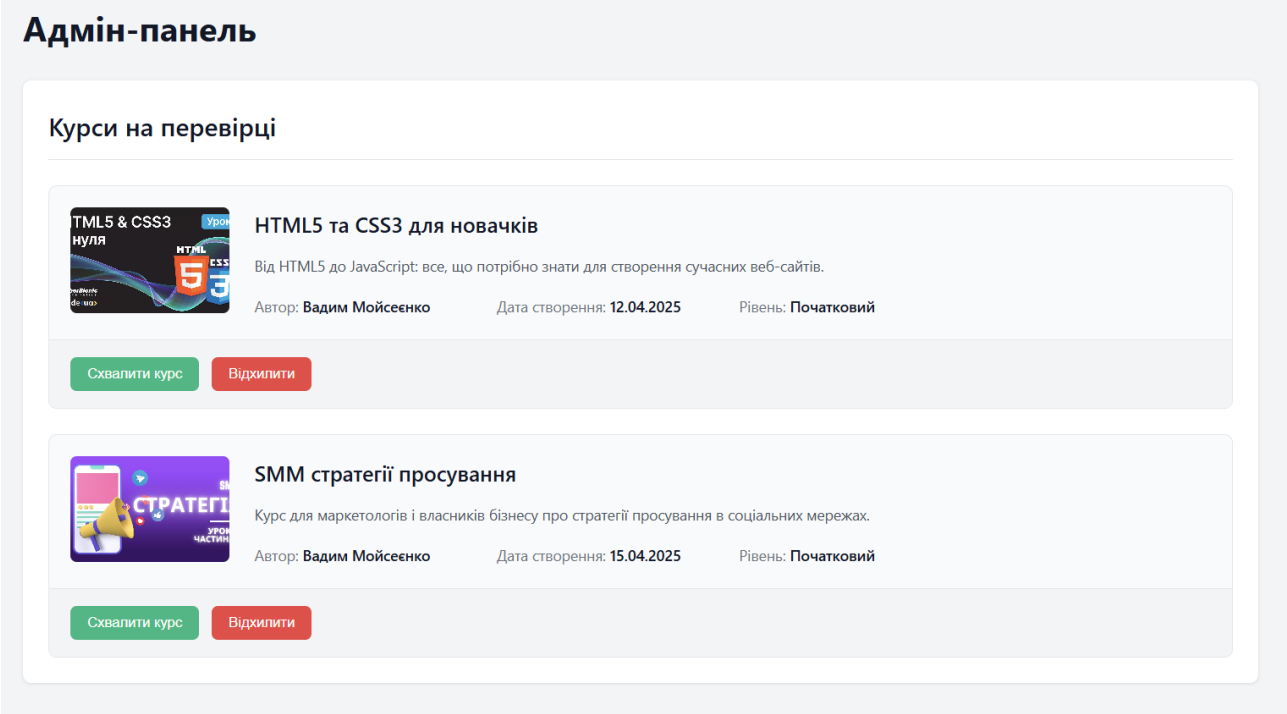


Рис. 4.21 Адміністративна панель для модераторів

У разі відхилення курсу модератор може вказати причину, яка буде відображена автору курсу для внесення відповідних змін.

4.4 Розробка тестових сценаріїв

Для забезпечення якості та надійності web-платформи Codegram було розроблено набір ключових тестових сценаріїв, які охоплюють основні функціональні можливості системи. Ці сценарії фокусуються на найважливіших аспектах роботи платформи та дозволяють ефективно перевірити її працездатність.

Основні тестові сценарії представлені в таблиці 4.1. Вони охоплюють критичні функціональні модулі системи: реєстрацію та вхід, роботу з каталогом курсів, проходження навчальних матеріалів, управління профілем користувача та створення курсів.

Основні тестові сценарії для ручного тестування web-платформи наведено у таблиці 4.1

Таблиця 4.1

Основні тестові сценарії для тестування web-платформи Codegram

№	Назва тестового сценарію	Кроки тестування	Очікуваний результат	Фактичний результат
1	Реєстрація нового користувача	1. Відкрити головну сторінку. 2. Натиснути кнопку "Приєднатися". 3. Заповнити поля. 4. Натиснути кнопку "Зареєструватися".	1. Відображається форма реєстрації. 2. Система приймає введені дані. 3. Створюється новий обліковий запис.	1. Форма реєстрації відображається коректно. 2. Дані приймаються після валідації. 3. Обліковий запис успішно створено.
2	Вхід існуючого користувача	1. Відкрити головну сторінку. 2. Натиснути кнопку "Увійти". 3. Ввести email та пароль. 4. Натиснути "Увійти".	1. Відображається форма входу. 2. Система перевіряє дані. 3. Користувач успішно авторизується.	1. Форма входу відображається коректно. 2. Система успішно перевіряє дані. 3. Авторизація проходить успішно.
3	Пошук та фільтрація курсів	1. Перейти на сторінку каталогу курсів. 2. Ввести пошуковий запит у поле пошуку. 3. Вибрати фільтри в бічній панелі.	1. Відображаються курси, що відповідають обраним фільтрам. 2. При зміні фільтрів результати оновлюються динамічно.	1. Результати пошуку відповідають запиту. 2. Фільтри працюють коректно. 3. Оновлення результатів відбувається коректно.
4	Запис на курс	1. Перейти на сторінку деталей курсу. 2. Натиснути кнопку "Записатись на курс".	1. Користувач додається до курсу. 2. Курс з'являється в розділі "Мої курси" в профілі користувача.	1. Запис на курс відбувається успішно. 2. Курс відображається в профілі.

Продовження таблиці 4.1

Основні тестові сценарії для тестування web-платформи Codegram

№	Назва тестового сценарію	Кроки тестування	Очікуваний результат	Фактичний результат
5	Перегляд та проходження уроку	1. Перейти на сторінку курсу. 2. Обрати урок зі списку. 3. Переглянути вміст уроку. 4. Натиснути "Позначити як виконане".	1. Відображається зміст уроку. 2. Урок позначається як виконаний. 3. Оновлюється прогрес курсу.	1. Урок успішно позначається як виконаний. 2. Прогрес оновлюється коректно.
6	Редагування профілю користувача	1. Перейти на сторінку профілю. 2. Натиснути "Редагувати профіль". 3. Змінити дані. 4. Натиснути "Зберегти зміни".	1. Відкривається вікно редагування. 2. Зміни зберігаються в базі даних. 3. Профіль оновлюється.	1. Модальне вікно відкривається правильно. 2. Дані зберігаються в базі. 3. Профіль оновлюється.
7	Створення нового курсу	1. Перейти на "Мої курси". 2. Натиснути "Створити новий курс". 3. Заповнити інформацію. 4. Створити тест. 5. Натиснути "Відправити на перевірку".	1. Створюється новий курс зі статусом "на перевірці". 2. Курс з'являється в розділі "На перевірці".	1. Курс успішно створюється. 2. Статус призначається правильно.
8	Модерація курсу	1. Увійти як модератор. 2. Відкрити адмін-панель. 3. Вибрати курс зі списку на перевірці. 4. Перевірити вміст курсу. 5. Натиснути "Схвалити" або "Відхилити".	1. Статус курсу змінюється на "схвалено" або "відхилено". 2. Схвалений курс з'являється в каталозі курсів.	1. Статус змінюється коректно. 2. Схвалений курс правильно відображається в каталозі.

Проведення тестування за цими сценаріями дозволяє забезпечити стабільну роботу критично важливих компонентів системи та швидко виявляти потенційні проблеми. Результати тестування використовуються для планування подальших вдосконалень платформи.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досягнуто поставленої мети – розроблено web-платформу Codegram для спрощення доступу до освітніх IT-курсів, яка забезпечує повний цикл освітнього процесу від створення до проходження навчальних матеріалів. У ході дослідження було досягнуто таких результатів:

1. Проведено аналіз існуючих онлайн-платформ для проходження IT-курсів (Prometheus, EdEra, Udemy, Coursera), що дозволило виявити їх ключові функціональні можливості, а також визначити переваги й недоліки кожної з них. Встановлено, що жодна з проаналізованих платформ не забезпечує поєднання україномовного інтерфейсу, можливості створення власних курсів користувачами та системи модерації контенту.

2. Досліджено сучасні web-технології та інструменти, обґрунтовано їх вибір для розробки web-платформи. Використання JavaScript як основної мови програмування, фреймворку Vue.js для розробки користувацького інтерфейсу, Pinia для управління станом додатку та Firebase як Backend-as-a-Service рішення забезпечило оптимальний баланс між швидкістю розробки та функціональністю системи.

3. Визначено перелік функціональних та нефункціональних вимог до розроблюваної системи, серед яких пріоритетними є можливість створення курсів та їх проходження з відстеженням прогресу.

4. Спроєктовано архітектуру web-платформи з використанням Firebase як серверної платформи, що забезпечує розмежування доступу відповідно до ролей користувачів, ефективне управління даними та масштабованість системи. Розроблено структуру бази даних, що включає взаємопов'язані колекції для зберігання інформації про користувачів, курси, модулі, уроки та прогрес навчання.

5. Реалізовано функціональність web-платформи на основі фреймворку Vue.js, що включає систему створення, модерації курсів та їх проходження з

відстеженням прогресу навчання. Взаємодія з серверною частиною організована через Firebase API, що забезпечує синхронізацію даних в режимі реального часу. Розроблено зручний та інтуїтивно зрозумілий інтерфейс користувача, який адаптується до різних розмірів екрану.

6. Проведено тестування web-платформи за розробленими тестовими сценаріями, яке підтвердило її відповідність функціональним та нефункціональним вимогам. Тестування показало, що система коректно виконує всі основні функції.

Отже, виконана кваліфікаційна робота демонструє успішну реалізацію web-платформи Codegram, яка ефективно вирішує актуальну проблему доступу до якісної IT-освіти в україномовному сегменті. Розроблена система забезпечує повний цикл освітнього процесу від створення контенту до його засвоєння з контролем якості через систему модерації. Створена платформа відрізняється від існуючих аналогів поєднанням українського інтерфейсу, можливості користувачів створювати власні курси та системою перевірки контенту, що свідчить про успішне досягнення поставленої мети та робить її цінним інструментом для розвитку у сфері IT-освіти.

ПЕРЕЛІК ПОСИЛАНЬ

1. Мойсеєнко В.Д., Яскевич В.О. Порівняльний аналіз фреймворків vue.js та react у сучасній веброзробці. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24.04.2025р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С. 131-133.

2. Мойсеєнко В.Д., Яскевич В.О. Переваги та недоліки використання бази даних firestore для розробки вебдодатків. VI Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15.04.2025 р., Київ, Державний університет інформаційно-комунікаційних технологій (подано до друку).

3. Талалаєв В., Денисюк А., Кириленко А. Використання функціональних шаблонів при створенні і розвитку відкритих онлайн платформ дистанційного навчання. *Modern engineering and innovative technologies*. 2024. С. 124–134. URL: https://doi.org/10.30890/2567-5273.2024-32-00-074.

4. Васильєва Т. А., Котенко С. І. Проблеми і перспективи розвитку онлайн-освіти. Суми, 2023. 124 с.

5. Морзе Н., Буйницька О. Модернізація освіти в цифровому вимірі. Київ, 2021. 299 с.

6. Лиходєєва Г. В., Хмельницька О. С., Київська К. І. Інноваційні технології в дистанційному навчанні: відкриті ресурси, онлайн-курси та інші можливості. *Академічні візії*. 2023. Випуск 20. URL: https://www.academy-vision.org/index.php/av/article/view/439.

7. Prometheus. URL: https://prometheus.org.ua.

8. EdEra. URL: https://ed-era.com.

9. Udemy. URL: https://www.udemy.com.

10. Coursera. URL: https://www.coursera.org.

11. MDN Web Docs: JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

12. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. 4-те вид. 2024. 435 с. URL: https://eloquentjavascript.net/Eloquent_JavaScript.pdf.

13. ECMAScript 2026 Language Specification. URL: https://tc39.es/ecma262.

14. Vue.js Documentation. Vue.js. URL: https://vuejs.org/guide/introduction.html.

15. Vue Composition API Documentation. Vue.js. URL: https://vuejs.org/guide/extras/composition-api-faq.html.

16. Vue Router Documentation. Vue.js. URL: https://router.vuejs.org/guide.

17. Vite.js: Next Generation Frontend Tooling. URL: https://vite.dev/guide.

18. Pinia: the intuitive store for Vue.js. URL: https://pinia.vuejs.org/introduction.html.

19. Firebase Documentation: Get started with Firebase for web. URL: https://firebase.google.com/docs/web/setup.

20. Cloud Firestore Documentation: Data model. URL: https://firebase.google.com/docs/firestore/data-model.

21. Firebase Security Rules Documentation. Google Firebase. URL: https://firebase.google.com/docs/rules.

22. Visual Studio Code Documentation: Setup overview. URL: https://code.visualstudio.com/docs/setup/setup-overview.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-платформи для надання доступу до освітніх IT-курсів

Виконав студент 4 курсу
групи ПД-41
Мойсеєнко Вадим Денисович
Керівник роботи
к.т.н., доц., доцент кафедри ІПЗ Яскевич Владислав Олександрович

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – спрощення доступу до освітніх IT-курсів шляхом розробки web-платформи для створення, модерації та проходження онлайн-курсів.
- **Об’єкт дослідження** – процес організації онлайн-навчання у сфері інформаційних технологій.
- **Предмет дослідження** – web-платформа Codegram для створення та проходження онлайн-курсів з функціями модерації та відстеження прогресу.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз існуючих онлайн-платформ для проходження IT-курсів, виявити їхні переваги та недоліки.
2. Дослідити сучасні технології та інструменти, які можуть бути використані для розробки web-платформи.
3. Сформулювати перелік функціональних та нефункціональних вимог до майбутньої платформи.
4. Спроєктувати архітектуру web-платформи для доступу до освітніх IT-курсів, що забезпечує функціонал створення, модерації та проходження онлайн-курсів.
5. Реалізувати програмну частину платформи відповідно до визначених вимог та спроєктованої архітектури.
6. Провести тестування розробленої системи.

3

АНАЛІЗ АНАЛОГІВ

Характеристика	Prometheus	EdEra	Udemy	Coursera	Codegram
Українська мова інтерфейсу	+	+	-	+	+
Велика різноманітність IT-напрямків	-	-	+	+	+
Можливість створення курсів користувачами	-	-	+	-	+
Підтримка мобільних пристроїв	+	+	+	+	+
Модерація користувацького контенту	-	-	+	-	+
Відстеження прогресу	+	+	+	+	+

4

ВИМОГИ ДО WEB-ПЛАТФОРМИ

Функціональні вимоги:

1. Реєстрація та авторизація користувачів.
2. Реалізація системи ролей із різними правами доступу (користувач, модератор).
3. Можливість пошуку та фільтрації курсів.
4. Можливість проходження курсів користувачами з відстеженням прогресу.
5. Можливість створення, редагування та видалення курсів авторами.
6. Можливість перегляду профілів та публікацій інших користувачів.
7. Адміністративна панель для управління курсами.

Нефункціональні вимоги:

1. Захист паролів користувачів (хешування за допомогою bcrypt).
2. Адаптивний дизайн інтерфейсу для різних розмірів екрану.
3. Швидке завантаження сторінок сайту (не більше 3 секунд).
4. Підтримка сучасних браузерів: Chrome, Firefox, Edge, Opera.

5

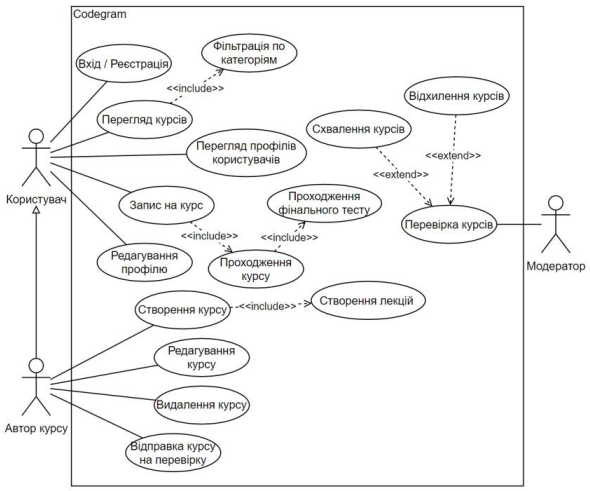
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Pinia

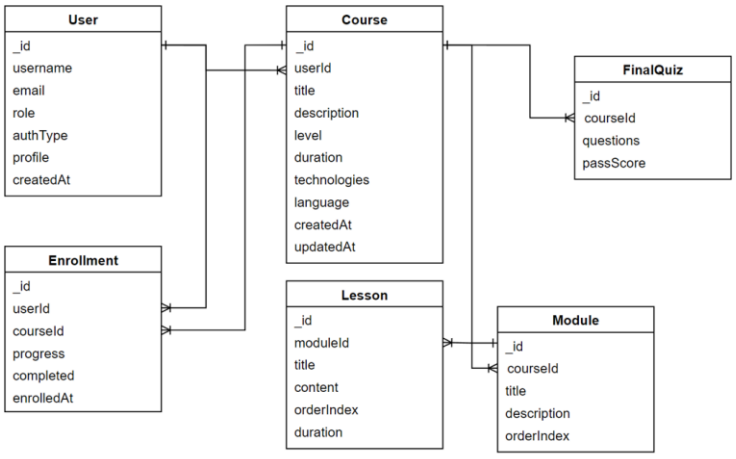
VS Code

6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

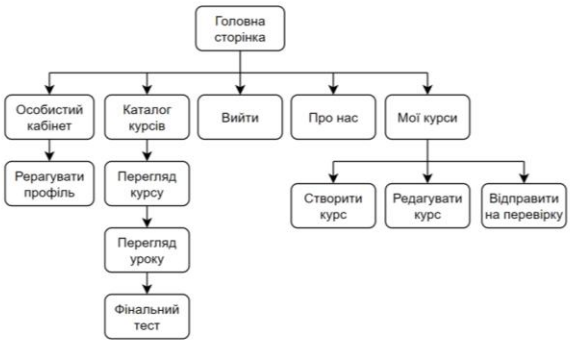


СТРУКТУРА БАЗИ ДАНИХ



МАПИ WEB-ПЛАТФОРМИ

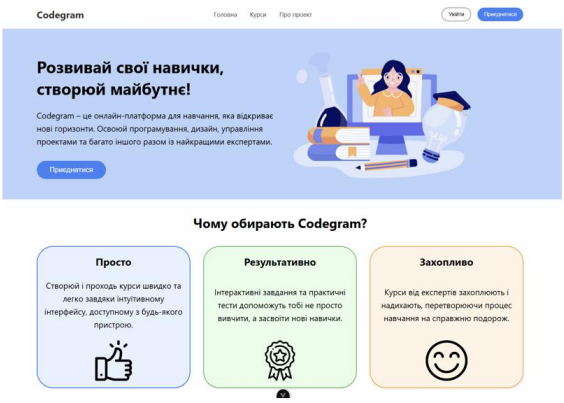
Мапа сайту неавторизованого користувача



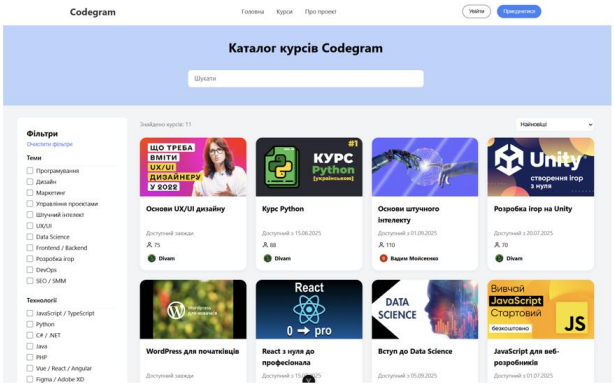
Мапа сайту авторизованого користувача

9

ЕКРАННІ ФОРМИ



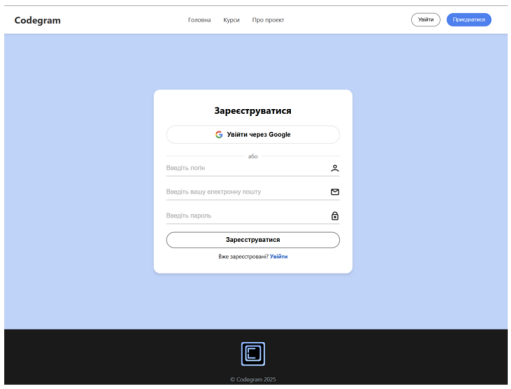
Головна сторінка сайту



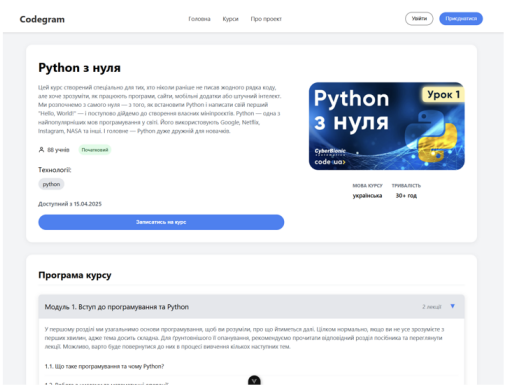
Каталог курсів

10

ЕКРАННІ ФОРМИ

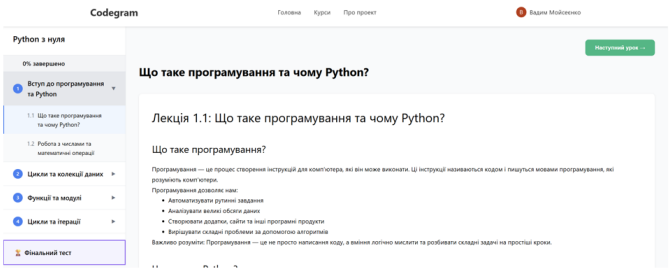


Реєстрація користувача

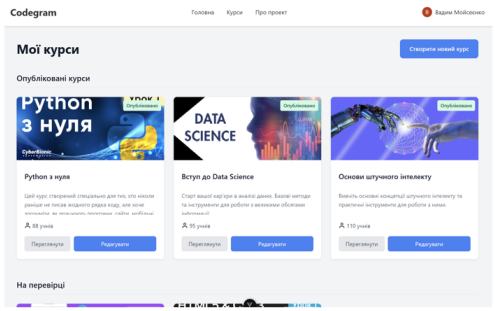


Інформація про курс

ЕКРАННІ ФОРМИ



Лекція курсу



Сторінка мої курси

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Мойсеєнко В.Д., Яскевич В.О. Порівняльний аналіз фреймворків vue.js та react у сучасній веброзробці. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24.04.2025р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С. 131-133.
2. Мойсеєнко В.Д., Яскевич В.О. Переваги та недоліки використання бази даних firestore для розробки вебдодатків. VI Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15.04.2025 р., Київ, Державний університет інформаційно-комунікаційних технологій (подано до друку).

ВИСНОВКИ

1. Проведено аналіз існуючих онлайн-платформ для проходження IT-курсів (Prometheus, EdEra, Udemy, Coursera), що дозволило виявити їх ключові функціональні можливості, а також визначити переваги й недоліки кожної з них.
2. Проаналізовано сучасні технології та інструменти, обґрунтовано їх вибір для розробки web-платформи.
3. Визначено функціональні та нефункціональні вимоги до розроблюваної системи, які включають можливість створення, модерацію та проходження онлайн-курсів з відстеженням прогресу навчання.
4. Спроектовано архітектуру web-платформи з використанням Firebase як серверної платформи, що забезпечує аутентифікацію, зберігання даних та розмежування доступу відповідно до ролей користувачів.
5. Реалізовано функціональність web-платформи на основі фреймворку Vue.js, що включає функціонал для створення навчальних курсів з модулями, уроками та фінальним тестом, систему модерації контенту та можливість проходження курсів з відстеженням прогресу навчання.
6. Проведено тестування web-платформи, яке підтвердило її відповідність функціональним та нефункціональним вимогам.

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

```
import { computed, reactive } from 'vue'
import { getFirestore, doc, getDoc, collection, getDocs,
query, orderBy } from 'firebase/firestore'
import { CoursesService } from
'../services/courses.service'

export function useCourseDetail(courseId) {
  const coursesService = new CoursesService()
  const db = getFirestore()

  // State management
  const state = reactive({
    course: {},
    author: null,
    modules: [],
    moduleLessons: {},
    loading: {
      course: true,
      lessons: {},
    },
    errors: {
      course: null,
      lessons: {},
    },
    expandedModuleIds: new Set(),
    showAllModules: false,
    isModuleClickable: true,
  })

  // Computed properties
  const sortedModules = computed(() =>
    [...state.modules].sort((a, b) => (a.order || 0) -
(b.order || 0)),
  )

  const isAvailable = computed(() => {
    const { availableFrom } = state.course

    if (!availableFrom) return true
    if (availableFrom === 'постійно') return true

    try {
      const [day, month, year] =
availableFrom.split('.').map(Number)
      if (isNaN(day) || isNaN(month) || isNaN(year))
return false

      const availableDate = new Date(year, month - 1,
day)
      return new Date() >= availableDate
    } catch (e) {
      console.error('Error parsing date:', e)
      return false
    }
  })

  // For template compatibility
  const fetchCourse = async () => {
    state.loading.course = true
    state.errors.course = null

    try {
      const courseRef = doc(db, 'courses', courseId)
      const courseDoc = await getDoc(courseRef)

      if (!courseDoc.exists()) {
        state.errors.course = 'Курс не знайдено'
        return
      }

      state.course = {
        id: courseDoc.id,
        ...courseDoc.data(),
      }

      await Promise.all([fetchAuthor(), fetchModules()])
    } catch (err) {
      console.error('Error fetching course details:', err)
      state.errors.course = `Помилка при завантаженні
курсу: ${err.message}`
    } finally {
      state.loading.course = false
    }
  }

  const fetchAuthor = async () => {
    if (!state.course.authorId) return null

    try {
      state.author = await
coursesService.getAuthorById(state.course.authorId)
    } catch (err) {
      console.error('Error fetching author:', err)
    }
  }

  const fetchModules = async () => {
    try {
      const modulesRef = collection(db,
`courses/${courseId}/modules`)
      const modulesQuery = query(modulesRef,
orderBy('order', 'asc'))
      const modulesSnapshot = await
getDocs(modulesQuery)

      const modulesData = []
      modulesSnapshot.forEach((moduleDoc) => {
        modulesData.push({
          id: moduleDoc.id,
          ...moduleDoc.data(),
        })
      })

      state.modules = modulesData

      if (modulesData.length > 0) {
```

```
      const firstModuleId = modulesData[0].id
      state.expandedModuleIds.add(firstModuleId)
      fetchLessonsForModule(firstModuleId)
    }

    state.showAllModules = modulesData.length <= 3
  } catch (err) {
    console.error('Error fetching modules:', err)
    state.errors.course = `Помилка при завантаженні модулів: ${err.message}`
  }
 }

 const fetchLessonsForModule = async (moduleId) => {
  if (state.moduleLessons[moduleId]) return

  state.loading.lessons[moduleId] = true
  state.errors.lessons[moduleId] = null

  try {
    const lessonsPath = `courses/${courseId}/modules/${moduleId}/lessons`
    const lessonsRef = collection(db, lessonsPath)
    const lessonsQuery = query(lessonsRef, orderBy('order', 'asc'))
    const lessonsSnapshot = await getDocs(lessonsQuery)

    const lessons = []
    lessonsSnapshot.forEach((lessonDoc) => {
      lessons.push({
        id: lessonDoc.id,
        ...lessonDoc.data(),
      })
    })

    state.moduleLessons[moduleId] = lessons
  } catch (err) {
    console.error(`Error fetching lessons for module ${moduleId}:`, err)
    state.errors.lessons[moduleId] = `Помилка при завантаженні лекцій: ${err.message}`
  } finally {
    state.loading.lessons[moduleId] = false
  }
 }

 const toggleModule = (moduleId) => {
  if (!state.isModuleClickable) return

  state.isModuleClickable = false

  if (state.expandedModuleIds.has(moduleId)) {
    state.expandedModuleIds.delete(moduleId)
  } else {
    state.expandedModuleIds.add(moduleId)
    fetchLessonsForModule(moduleId)
  }

  setTimeout(() => {
    state.isModuleClickable = true
  }, 300)
 }

 const expandAllModules = () => {
  state.showAllModules = true

  state.modules.forEach((module) => {
    if (!state.expandedModuleIds.has(module.id)) {
      state.expandedModuleIds.add(module.id)
      fetchLessonsForModule(module.id)
    }
  })
 }

 const formatAvailability = (availability) => {
  if (!availability) return 'Доступність не вказана'
  if (availability === 'постійно') return 'Доступний завжди'
  return `Доступний з ${availability}`
 }

 const formatLevel = (level) => {
  const levels = {
    beginner: 'Початковий',
    intermediate: 'Середній',
    advanced: 'Просунутий',
  }
  return levels[level] || level
 }
}

export class AuthHelper {
 constructor() {
  this.db = getFirestore()
  this.auth = getAuth()
 }

 // Check if an email exists in the database
 async checkEmailExists(email) {
  if (!email) {
    return { exists: false, authType: null }
  }

  try {
    const normalizedEmail = email.toLowerCase()
    const usersRef = collection(this.db, 'users')
    const q = query(usersRef, where('email', '==', normalizedEmail), limit(1))
    const querySnapshot = await getDocs(q)

    if (!querySnapshot.empty) {
      const userData = querySnapshot.docs[0].data()
      return {
        exists: true,
        authType: userData.authType,
      }
    }

    return {
      exists: false,
      authType: null,
    }
```

```
    } catch (error) {
      console.error('Error checking email:', error)
      throw new Error(`Помилка перевірки email:
${error.message}`)
    }
  }

  // Check if a username exists in the database
  async checkUsernameExists(username) {
    if (!username) {
      return false
    }

    try {
      const usersRef = collection(this.db, 'users')

      const normalizedUsername =
username.toLowerCase()
      const q = query(usersRef,
where('normalizedUsername', '==',
normalizedUsername), limit(1))
      const querySnapshot = await getDocs(q)

      return !querySnapshot.empty
    } catch (error) {
      console.error('Error checking username:', error)
      throw new Error(`Помилка перевірки логіна:
${error.message}`)
    }
  }

  getUserEmail(user) {
    if (user.email) {
      return user.email
    }

    if (user.providerData && user.providerData.length >
0) {
      const googleData =
user.providerData.find((provider) =>
provider.providerId === 'google.com')
      if (googleData && googleData.email) {
        return googleData.email
      }
    }

    throw new Error('Не вдалося отримати email
користувача')
  }

  // Create a new user record in the database
  async createUserRecord(user, username, authType) {
    try {
      const email = this.getUserEmail(user)

      if (!email) {
        throw new Error("Email є обов'язковим для
створення профілю")
      }

      const finalUsername = username ||
email.split('@')[0]

      let avatarUrl = ''
      if (user.photoURL) {
        avatarUrl = user.photoURL.split('=')[0]
      }

      const userDoc = {
        userId: user.uid,
        email: email.toLowerCase(),
        username: finalUsername,
        normalizedUsername:
finalUsername.toLowerCase(),
        role: 'user',
        authType: authType,
        createdAt: new Date(),
        profile: {
          avatarUrl: avatarUrl,
          bio: '',
        },
      }

      const userRef = doc(this.db, 'users', user.uid)
      await setDoc(userRef, userDoc)

      return userDoc
    } catch (error) {
      console.error('Error in createUserRecord:', error)
      throw new Error(`Не вдалося створити профіль
користувача: ${error.message}`)
    }
  }

  // Register a new user with email and password
  async registerWithEmail(email, password, username)
{
    try {
      const { exists: emailExists, authType } = await
this.checkEmailExists(email)

      if (emailExists) {
        if (authType === 'email') {
          throw new Error('Користувач з такою
електронною поштою вже існує')
        }
        throw new Error('Ця електронна пошта вже
використовується через інший метод
автентифікації')
      }

      if (username) {
        const usernameExists = await
this.checkUsernameExists(username)
        if (usernameExists) {
          throw new Error('Цей логін вже
використовується іншим користувачем')
        }
      }

      const userCredential = await
createUserWithEmailAndPassword(this.auth, email,
password)
```

```
      await this.createUserRecord(userCredential.user,
username, 'email')

      return userCredential.user
    } catch (error) {
      console.error('Registration error:', error)

      if (error.code === 'auth/email-already-in-use') {
        throw new Error('Ця електронна пошта вже
використовується')
      }
      if (error.code === 'auth/invalid-email') {
        throw new Error('Невірний формат електронної
пошти')
      }
      if (error.code === 'auth/weak-password') {
        throw new Error('Пароль занадто слабкий')
      }

      if (error.code?.startsWith('auth/')) {
        throw new Error(`Помилка реєстрації:
${error.message}`)
      }

      if (error.message) {
        throw error
      }

      throw new Error('Помилка при реєстрації.
Спробуйте ще раз.')
    }
  }

  // Sign in with email and password
  async signInWithEmail(email, password,
rememberMe = false) {
    try {
      const { exists, authType } = await
this.checkEmailExists(email)

      if (exists && authType !== 'email') {
        throw new Error('Цей email зареєстрований
через інший метод автентифікації')
      }

      const persistence = rememberMe ?
browserLocalPersistence : browserSessionPersistence
      await setPersistence(this.auth, persistence)

      const userCredential = await
signInWithEmailAndPassword(this.auth, email,
password)
      return userCredential.user
    } catch (error) {
      console.error('Login error:', error)

      if (error.code === 'auth/invalid-email') {
        throw new Error('Невірний формат електронної
пошти')
      }
      if (error.code === 'auth/user-not-found') {
        throw new Error('Користувача з такою
електронною поштою не знайдено')
      }
      if (error.code === 'auth/wrong-password') {
        throw new Error('Невірний пароль')
      }
      if (error.code === 'auth/too-many-requests') {
        throw new Error('Забагато спроб входу.
Спробуйте пізніше')
      }
      if (error.code === 'auth/invalid-credential') {
        throw new Error('Невірні облікові дані.
Перевірте email та пароль')
      }

      if (error.code?.startsWith('auth/')) {
        throw new Error(`Помилка входу:
${error.message}`)
      }

      if (error.message) {
        throw error
      }

      throw new Error('Помилка при вході. Спробуйте
ще раз.')
    }
  }

  // Sign in with Google
  async signInWithGoogle() {
    let temporaryUser = null

    try {
      const provider = new GoogleAuthProvider()
      provider.addScope('email')

      const result = await signInWithPopup(this.auth,
provider)
      temporaryUser = result.user

      const email = this.getUserEmail(result.user)

      const { exists, authType } = await
this.checkEmailExists(email)

      if (exists && authType !== 'google') {
        await temporaryUser.delete()
        throw new Error('Цей email вже зареєстрований
через інший метод автентифікації')
      }

      if (!exists) {
        await this.createUserRecord(result.user,
result.user.displayName, 'google')
      }

      return result.user
    } catch (error) {
      console.error('Google sign in error:', error)

      if (temporaryUser) {
```

```
      try {
        await temporaryUser.delete()
      } catch (deleteError) {
        console.error('Error deleting temporary user:',
deleteError)
      }
    }

    throw error
  }
 }

 // Send password reset email
 async sendPasswordReset(email) {
  try {
    const { exists, authType } = await
this.checkEmailExists(email)

    if (!exists) {
      throw new Error('Користувача з такою
електронною поштою не знайдено')
    }

    if (authType !== 'email') {
      throw new Error('Цей email зареєстрований
через Google. Використовуйте Google для входу.')
    }

    await sendPasswordResetEmail(this.auth, email)

    return true
  } catch (error) {
    console.error('Password reset error:', error)

    if (error.code === 'auth/invalid-email') {
      throw new Error('Невірний формат електронної
пошти')
    }
    if (error.code === 'auth/user-not-found') {
      throw new Error('Користувача з такою
електронною поштою не знайдено')
    }

    if (error.code?.startsWith('auth/')) {
      throw new Error(`Помилка відновлення
паролю: ${error.message}`)
    }

    if (error.message) {
      throw error
    }

    throw new Error('Помилка при відновленні
паролю. Спробуйте ще раз.')
  }
 }
}

export class QuizzesService {
 constructor() {
  this.db = getFirestore()
 }

 async getFinalQuiz(courseId) {
  try {
    const finalQuizCollection = collection(this.db,
`courses/${courseId}/finalQuiz`)
    const finalQuizSnapshot = await
getDocs(finalQuizCollection)

    if (!finalQuizSnapshot.empty) {
      const quizDoc = finalQuizSnapshot.docs[0]

      return {
        id: quizDoc.id,
        ...quizDoc.data(),
        isFinalQuiz: true,
      }
    }

    const courseRef = doc(this.db, 'courses', courseId)
    const courseDoc = await getDoc(courseRef)

    if (courseDoc.exists() &&
courseDoc.data().finalQuiz) {
      const quizData = courseDoc.data().finalQuiz
      return {
        id: 'embedded-final-quiz',
        ...quizData,
        isFinalQuiz: true,
      }
    }

    return null
  } catch (error) {
    console.error(`Помилка при отриманні
фінального тесту для курсу ${courseId}:`, error)
    throw new Error(`Помилка при отриманні
фінального тесту: ${error.message}`)
  }
 }

 async saveQuizResult(enrollmentId, quizResult) {
  try {
    const quizData = {
      ...quizResult,
      completedAt: serverTimestamp(),
      isFinalQuiz: true,
    }

    // Перевіряємо, чи вже є результат для цього
тесту
    const existingResults = await
this.getQuizResults(enrollmentId)
    const existingResult = existingResults.find((result)
=> result.isFinalQuiz)

    if (existingResult) {
      // Оновлюємо існуючий результат
      const resultRef = doc(this.db,
`enrollments/${enrollmentId}/quizResults`,
existingResult.id)

      const updatedData = {
```

```
        ...quizData,
        attempts: (existingResult.attempts || 0) + 1,
       }

       await updateDoc(resultRef, updatedData)

       return {
        id: existingResult.id,
        ...updatedData,
       }
      } else {
       const resultData = {
        ...quizData,
        attempts: 1,
       }

       const resultsRef = collection(this.db,
`enrollments/${enrollmentId}/quizResults`)
       const newResultRef = await addDoc(resultsRef,
resultData)

       return {
        id: newResultRef.id,
        ...resultData,
       }
      }
    } catch (error) {
     console.error('Error saving quiz result:', error)
     throw new Error(`Помилка при збереженні
результату тесту: ${error.message}`)
    }
   }

   async getQuizResults(enrollmentId) {
    try {
     const resultsRef = collection(this.db,
`enrollments/${enrollmentId}/quizResults`)
     const resultsSnapshot = await getDocs(resultsRef)

     const results = []
     resultsSnapshot.forEach((doc) => {
      results.push({
       id: doc.id,
       ...doc.data(),
      })
     })

     return results
    } catch (error) {
     console.error('Error fetching quiz results:', error)
     throw new Error(`Помилка при отриманні
результатів тестів: ${error.message}`)
    }
   }

   async updateCourseCompletionStatus(enrollmentId) {
    try {
     const enrollmentRef = doc(this.db, 'enrollments',
enrollmentId)
     const enrollmentDoc = await
getDoc(enrollmentRef)

     if (!enrollmentDoc.exists()) {
      throw new Error('Запис на курс не знайдено')
     }

     // Отримуємо результат фінального тесту
     const results = await
this.getQuizResults(enrollmentId)
     const finalQuizResult = results.find((result) =>
result.isFinalQuiz)

     if (!finalQuizResult) {
      return {
       id: enrollmentId,
       completed: false,
       finalScore: 0,
      }
     }

     await updateDoc(enrollmentRef, {
      completed: passed,
      finalScore: finalQuizResult.score,
      completedAt: passed ? serverTimestamp() : null,
     })

     return {
      id: enrollmentId,
      completed: passed,
      finalScore: finalQuizResult.score,
     }
    } catch (error) {
     console.error('Error updating course completion
status:', error)
     throw new Error(`Помилка при оновленні
статусу проходження курсу: ${error.message}`)
    }
   }
}
```