

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

**«Онлайн-сервіс проходження курсів української
мови»**

Виконав студент групи ПЗ-19-2
СВІСТЕЛЬНИК Олександр Сергійович

Керівник роботи:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2023

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

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії

програмного забезпечення

Андрій МОРОЗОВ

«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: СВІСТЕЛЬНИК Олександр Сергійович

Керівник роботи: ЛОКТИКОВА Тамара Миколаївна

Тема роботи: «Онлайн-сервіс проходження курсів української мови»,
затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с

Вихідні дані для роботи: Об'єктом дослідження є використання інформаційних технологій для покращення рівня знань української мови за допомогою онлайн-сервісу.

Предметом дослідження є використання сучасних та веборієнтованих технологій розробки для створення вебзастосунку, що допомагає покращити рівень знань української мови.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	16.02.2023р.	16.02.2023р.
2	Локтікова Т.М.	16.02.2023р.	16.02.2023р.
3	Локтікова Т.М.	16.02.2023р.	16.02.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання	16.02.2023-16.03.2023	Виконано
2	Проектування структури системи	16.03.2023-16.04.2023	Виконано
3	Написання програмного коду	16.04.2023-01.06.2023	Виконано

4	Тестування	01.06.2023-07.06.2023	Виконано
5	Оформлення пояснювальної записки	07.06.2023-14.06.2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу онлайн-сервісу проходження курсів української мови та пояснювальної записки.

Пояснювальна записка до випускної роботи містить 75 сторінок, 42 ілюстрації, 17 таблиць, 4 додатки та 26 літературних джерел.

Метою роботи є розробка онлайн-сервісу проходження курсів української мови для покращення знань громадян України.

В роботі визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи. Обґрунтовано використання архітектури додатку. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи, опис об'єктної структури системи (діаграма класів). Також наводиться результати тестування системи.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, API, LARAVEL, REACT, SPA.

					ІПЗ.КР.Б – 121 – 23 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		О.С. Свістельник			Онлайн-сервіс проходження курсів української мови	Літ.	Арк.	Аркуші	
Керівник		Т.М. Локтікова					4	75	
						Житомирська політехніка, група ІПЗ-19-2			
Рецензент		А.В. Морозов							
Зав. каф.		А.В. Морозов							
					Пояснювальна записка				

ABSTRACT

The bachelor's thesis consists of the software complex of the online service for taking Ukrainian language courses and an explanatory note.

The explanatory note to the final thesis contains 75 pages, 42 illustrations, 17 tables, 4 appendices and 26 literary sources.

The purpose of the work is to develop an online service for taking Ukrainian language courses to improve the knowledge of Ukrainian citizens.

The paper defines the main tasks for the development of the system, and analyzes the analogues of the developed system. The use of the application architecture is justified. Scenarios of the software complex, the general structure of the software complex, a description of the interaction algorithms of individual system modules, a description of the object structure of the system (class diagram) are given. The results of system testing are also provided.

KEY WORDS: API, LARAVEL, REACT, SPA, WEB APPLICATION.

					ІІІЗ.КР.Б – 121 – 23 – ІІЗ	
						5

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ОНЛАЙН-СЕРВІСУ ПРОХОДЖЕННЯ КУРСІВ УКРАЇНСЬКОЇ МОВИ.....	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури онлайн-сервісу	14
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	18
Висновки до першого розділу.....	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ ОНЛАЙН-СЕРВІСУ ПРОХОДЖЕННЯ КУРСІВ УКРАЇНСЬКОЇ МОВИ	22
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	22
2.2. Розробка бази даних системи.....	27
2.3. Проектування та реалізація алгоритмів роботи системи	33
2.4. Реалізація онлайн-сервісу проходження курсів української мови	37
Висновки до другого розділу	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН-СЕРВІСОМ ПРОХОДЖЕННЯ КУРСІВ УКРАЇНСЬКОЇ МОВИ	42
3.1. Порядок встановлення та налаштування параметрів системи	42
3.2. Структура інтерфейсу та порядок взаємодії з онлайн-сервісом	43
3.3. Тестування роботи програмного продукту	52
Висновки до третього розділу	55
ВИСНОВКИ.....	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	58
ДОДАТКИ.....	61

ВСТУП

Актуальність теми. У сучасному світі важко уявити людину, яка не використовує Інтернет. Щодня ми чуємо нові слова, значення яких нам невідомі, і ми одразу користуємося можливостями Всесвітньої мережі для отримання експертної думки з цього приводу. Ми можемо використовувати потужності Інтернету як нам заманеться, але чому б не спрямувати їх у правильне русло для нашої країни. На жаль, історично склалося так, що істотна кількість населення України не розмовляє державною мовою, надаючи перевагу спілкуватися мовою окупанта, який століттями намагався нас завоювати. Проте повномасштабне вторгнення російської федерації у лютому 2022 року докорінно змінило думку більшості свідомих громадян України про важливість мови для самоідентифікації держави в світовому суспільстві. Тому й було вирішено розробити вебзастосунок, який може полегшити процес “лагідної українізації” для мешканців України, які невпевнені у своїх знаннях державної мови.

Метою випускної роботи є розробка онлайн-сервісу проходження курсів української мови.

Дана мета передбачає виконання наступних завдань:

- проведення аналізу вимог до онлайн-сервісу;
- аналіз аналогів програмного продукту;
- вибір архітектури системи;
- обґрунтування та вибір засобів реалізації;
- визначення варіантів використання та проектування об’єктоорієнтованої структури системи;
- проектування алгоритмів системи;
- безпосередня реалізація застосунку;
- опис структури інтерфейсу та порядку взаємодії зі системою;
- тестування розробленого додатку.

Об’єктом дослідження є використання інформаційних технологій для покращення рівня знань української мови за допомогою онлайн-сервісу.

Предметом дослідження є використання сучасних веборієнтованих технологій розробки для створення вебзастосунку, що допомагає покращити рівень знань української мови.

Методи дослідження. У роботі було використано методи об'єктно-орієнтованого проєктування, об'єктно-орієнтованого та функціонального програмувань.

Практичне значення одержаних результатів. Розроблений вебзастосунок може використовуватися в навчальних цілях для покращення знань пересічних громадян України.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ОНЛАЙН-СЕРВІСУ ПРОХОДЖЕННЯ КУРСІВ УКРАЇНСЬКОЇ МОВИ

1.1. Постановка задачі

Розробка онлайн-сервісу проходження курсів української мови передбачає створення фронтенд частини застосунку за допомогою бібліотеки React [1] мови програмування JavaScript [2] та бекенд частини за допомогою фреймворку Laravel [3] мови програмування PHP [4].

Отриманий застосунок повинен бути вебсайтом, що надає можливість користувачу покращити знання української мови шляхом проходження курсів.

Програмний продукт повинен мати наступні функціональні характеристики:

- авторизація та реєстрація користувачів з можливістю скидання і зміни пароля;
- авторизація та реєстрація користувачів за допомогою Google-акаунту;
- сортування і фільтрація курсів за певними критеріями;
- пошук курсів за назвою;
- перегляд профілю і курсів, які було розпочато;
- перегляд розділу курсу з відео-контентом та опорним конспектом;
- додавання і зміна коментаря після проходження курсу;
- проходження тесту після опрацювання матеріалів курсу;
- оплата платного курсу відразу за допомогою застосунку.

Програмний комплекс повинен за допомогою клієнт-серверної архітектури з використанням веб-технологій, що не вимагають додаткового ліцензування. Серверна частина програмного комплексу має базуватися на веб-сервері Apache2 та системі керування базами даних MySQL [5]. При проектуванні програмного комплексу необхідно застосувати архітектуру SPA (Single-Page Application) [6]

для забезпечення користувачу досвіду, близького до користування настільною програмою.

Клієнтська частина повинна бути побудована за сучасними стандартами створення односторінкового застосунку, тобто весь необхідний код завантажується разом зі сторінкою, або динамічно довантажується за потребою, зазвичай у відповідь на дії користувача. Сторінка не оновлюється і не перенаправляє користувача до іншої сторінки у процесі роботи з нею.

Розробка системи передбачає наступні основні етапи:

1. Створення бекенд частину додатку шляхом написання приватного API [7], використовуючи фреймворк Laravel, що включає реалізацію наступних сутностей:
 - сутність “Курс”;
 - сутність “Тег”;
 - сутність “Розділ курсу”;
 - сутність “Авторизація”;
 - сутність “Тест”;
 - сутність “Оплата”;
 - сутність “Профіль користувача”;
 - сутність “Завантаження великих файлів”.
2. Створення дизайну інтерфейсу, з яким будуть взаємодіяти користувачі в системі:
 - обрати бібліотеку для реалізації користувацького інтерфейсу;
 - розробити відповідні сторінки за допомогою обраної бібліотеки;
 - передбачити сповіщення користувача про завантаження контенту за допомогою відповідного елементу.
3. Налаштувати отримання даних від створеного API на клієнті та розробити роутинг застосунку за допомогою бібліотеки React:
 - налаштувати реалізацію виклику API за допомогою відповідної бібліотеки;

- налагодити роутинг додатку за допомогою відповідної бібліотеки;
- передбачити розбиття відео на шматки для завантаження його на сервер за допомогою відповідної бібліотеки.

Результатом виконання цих етапів є створення системи, побудованої за архітектурою SPA (Single-Page Application), яка повинна надати користувачу зручну платформу для навчання.

1.2. Аналіз аналогів програмного продукту

В сучасному світі важко придумати щось нове, що досі не було придумано. Тому під час створення додатків необхідно звертати уваги на аналоги для отримання цілісної картини ринку та вимог користувачів до цих застосунків. Для аналізу було обрано наступні додатки:

1. Мова – ДНК нації [8].
2. UdeMy [9].
3. Є-Мова [10].

Мова – ДНК нації – це проєкт, що має інтерактивний сайт, та пропонує вивчати мову через різні інструменти.

На рис. 1.1 зображено перелік наявних тем.

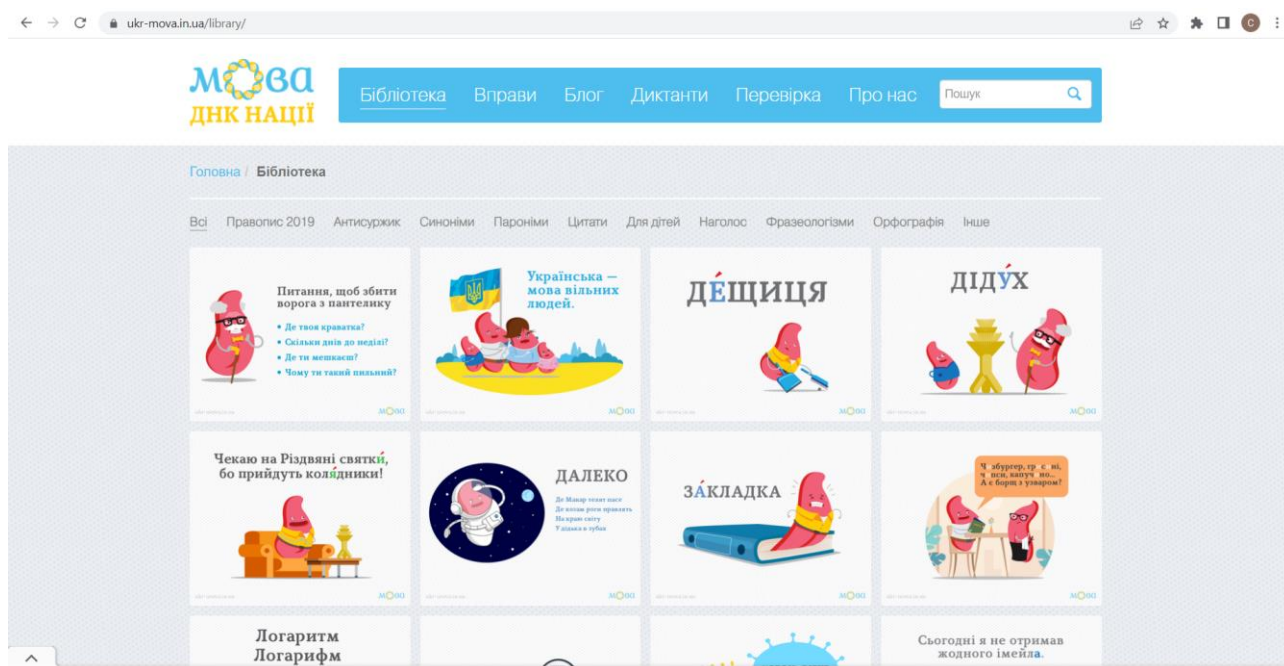


Рисунок 1.1 - Мова – ДНК нації (перелік тем)

Можна виокремити наступні переваги даного вебсайту:

- гарний та інтуїтивний дизайн;
- наявні коментарі;
- є розподіл за категоріями;
- наявні вправи для перевірки своїх знань.

До недоліків можна віднести наступне:

- відсутній відеоконтент;
- вузько направлена тематика окремих розділів.

Udemy - це онлайн-платформа, яка надає можливість проходження курсів з різних тематик. Вона пропонує величезний вибір курсів, що охоплюють різні сфери, зокрема і українську мову.

На рис. 1.2 зображено сторінку окремого курсу.

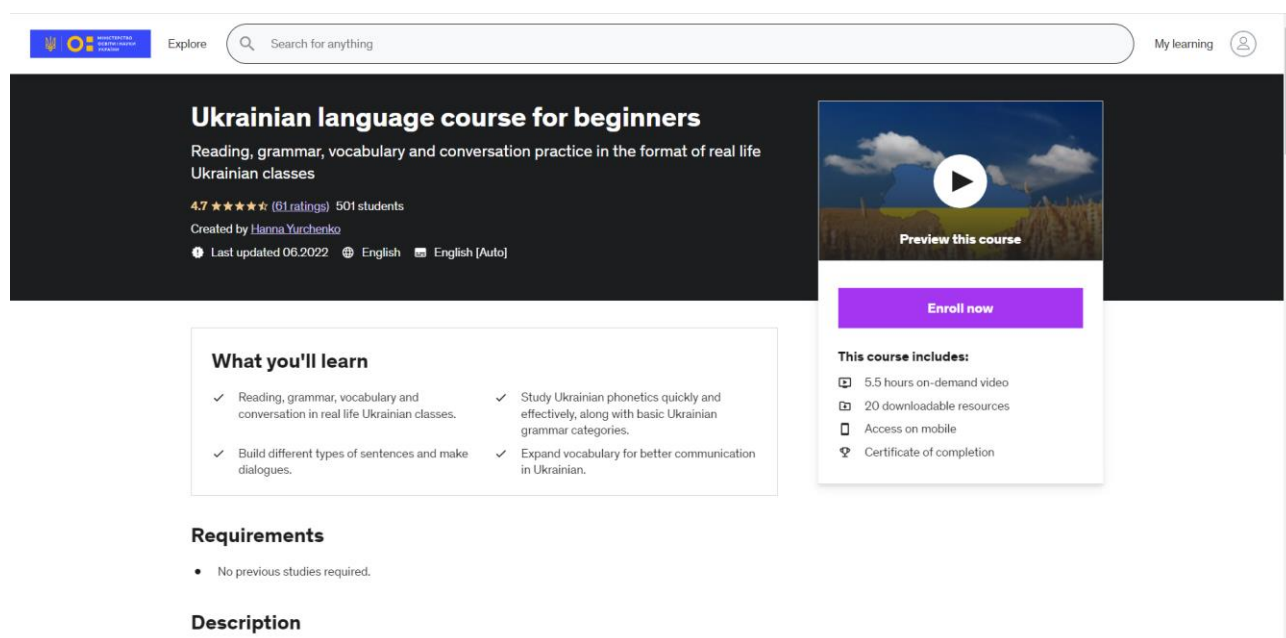


Рисунок 1.2 - Udemy (курс української мови)

До переваг можна віднести:

- гарний та інтуїтивний дизайн;
- наявні коментарі;
- зручний відеоплеєр.

Недоліками є наступне:

- відсутня українська мова на сайті;

- невелика кількість курсів на задану тему;
- відсутні вправи для перевірки знань.

Є-мова – це освітня платформа громадської організації “Український світ”, яка має на меті створити українськомовний культурний простір. На цій платформі ви можете знайти курси як з вивчення та популяризації української мови.

На рис. 1.3 зображено сторінку окремого курсу.

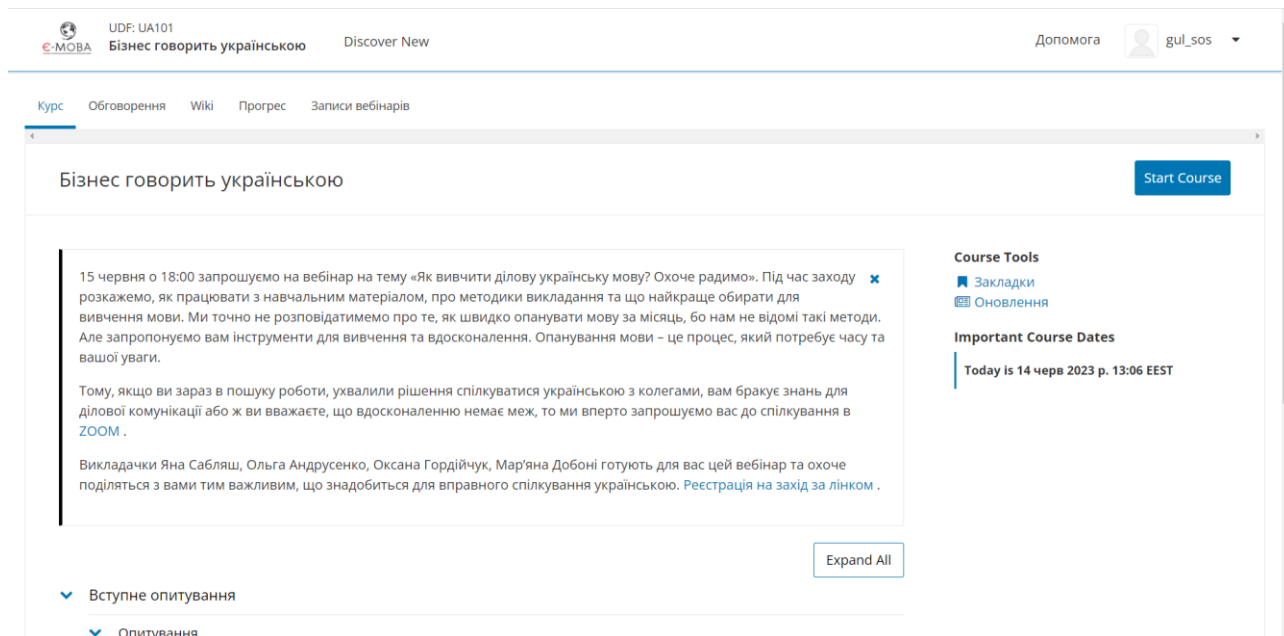


Рисунок 1.3 - Є-мова (обраний курс)

Перевагами є наступне:

- наявні коментарі;
- наявний відеоконтент;
- існують вправи для перевірки своїх знань.

До недоліків можна віднести наступне:

- має досить значні недоліки в дизайні та у вигляді загалом;
- вузько направлена тематика окремих курсів.

Структуруємо отримані дані за допомогою таблиці.

Таблиця 1.1

Характеристика	Мова – ДНК нації	Udemy	Є-мова
Інтуїтивний та зрозумілий дизайн	+	+	-
Наявність відеоконтенту	-	+	+
Наявність коментарів	+	+	+
Українська мова інтерфейсу	+	-	+
Достатня кількість курсів	+	-	+
Розподіл за категоріями	+	+	-
Наявність вправ для перевірки своїх знань	+	-	+

Грунтуючись на отриманих результатах, отримуємо наступні критерії для додатку, що розроблюється:

- інтуїтивний та зрозумілий дизайн;
- наявність відеоконтенту;
- наявність коментарів;
- українська мова інтерфейсу;
- достатня кількість курсів;
- розподіл за категоріями;
- наявність вправ для перевірки своїх знань.

1.3. Вибір архітектури онлайн-сервісу

В сучасному світі існує незліченна кількість архітектур для побудови вебдодатку. Розглянемо декілька з них, такі як:

- схема MVC (Model-View-Controller) [11];
- схема MVVM (Model-View-ViewModel) [12];
- схема SPA (Single-Page Application).

Тепер докладніше розберемо кожен схему.

MVC – це архітектурний шаблон, який дозволяє розділити компоненти додатку на три основні частини: модель (дані), вид (інтерфейс користувача) і

контролер (бізнес-логіка). Модель представляє дані додатку, вид відповідає за відображення інформації користувачу, а контролер обробляє запити користувача та взаємодіє з моделлю і видом.

На рис. 1.4 зображено схему MVC.

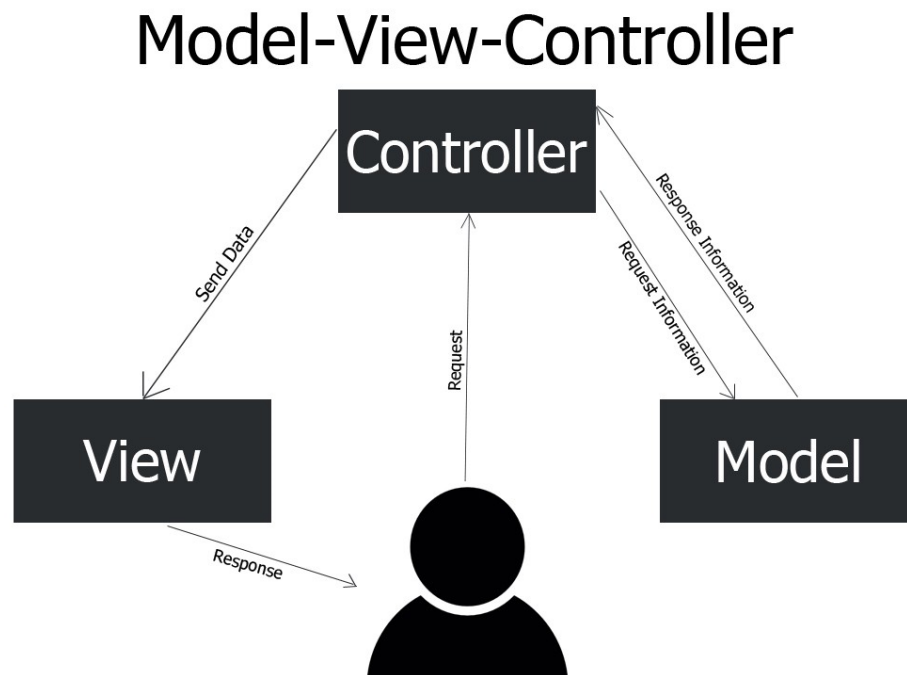


Рисунок. 1.4 - Схема MVC

До його переваг належать:

- розділення обов’язків: дозволяє чітко розділити логіку додатку на три основні компоненти – модель, вид і контролер. Це полегшує розуміння і підтримку коду, спрощує тестування і забезпечує більшу модульність;
- швидкість розробки: Завдяки розділенню відповідальностей між різними компонентами, команда розробників може паралельно працювати над різними аспектами додатку, що пришвидшує процес розробки;
- підтримка одночасної розробки: дозволяє різним членам команди працювати над окремими компонентами, незалежно один від одного. Це дозволяє швидко розробляти проект і легко масштабувати команду розробників.

До недоліків можна віднести наступне:

- зайва складність для простих додатків: Для дуже простих додатків або проєктів з невеликим функціоналом, використання повного MVC може бути зайвим. Це може призвести до зайвого навантаження і збільшення складності коду;
- залежність від контролера: В деяких випадках контролер може стати централізованою точкою в додатку і надмірно навантажити з логічної точки зору. Це може вплинути на швидкість і продуктивність додатку.

MVVM – це архітектурний шаблон, який розділяє додаток на три основні частини: модель (дані), вид (інтерфейс користувача) і вид-модель (логіка взаємодії між моделлю і видом). Вид-модель відповідає за обробку даних з моделі та підготовку їх для відображення в виді.

На рис. 1.5 зображено схему MVVM.

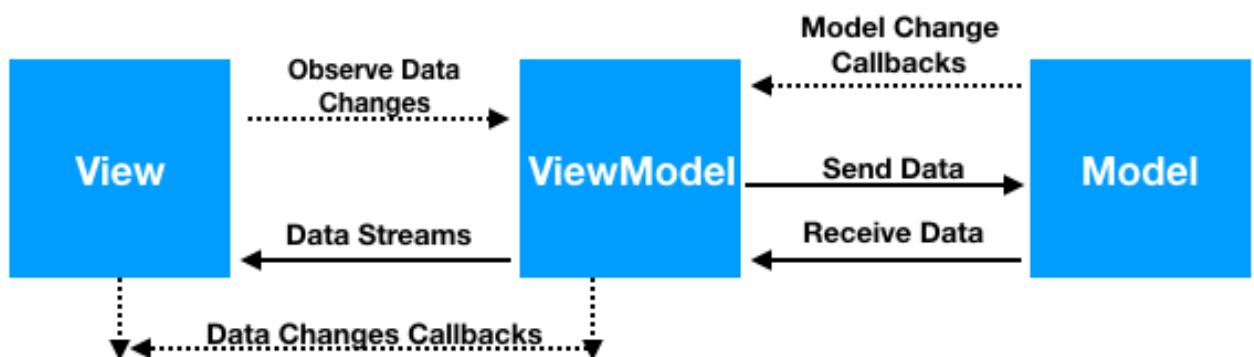


Рисунок 1.5 - Схема MVVM

Можна виділити наступні переваги для даної архітектури:

- декларативний підхід: використовується декларативний підхід до програмування, що означає, що ви описуєте, як ваше відображення має виглядати в залежності від даних вид-моделі. Це спрощує розробку і підтримку інтерфейсу користувача;
- прив'язка даних: використовується двосторонню прив'язку даних, що означає, що зміни в моделі автоматично відображаються в вид-моделі, а зміни у вид-моделі автоматично оновлюють відображення виду. Це дозволяє забезпечити синхронізацію між даними і відображенням.

До недоліків належать:

- навантаження пам'яті: MVVM може призвести до додаткового навантаження пам'яті через зберігання додаткових об'єктів вид-моделі. Це може бути проблемою для додатків з великою кількістю даних або для пристроїв з обмеженими ресурсами;
- залежність від фреймворка: Деякі функціональності можуть вимагати використання спеціальних фреймворків або бібліотек. Це може створити залежність від конкретного інструменту і обмежити гнучкість розробки.

SPA – це архітектурний підхід, в якому весь веб-додаток завантажується одним разом, а взаємодія з сервером відбувається через асинхронні запити (наприклад, за допомогою AJAX). Цей підхід дозволяє створити більш різноманітний інтерфейс, зменшити завантаження сторінок і збільшити продуктивність додатку.

На рис. 1.6 зображено схему SPA.

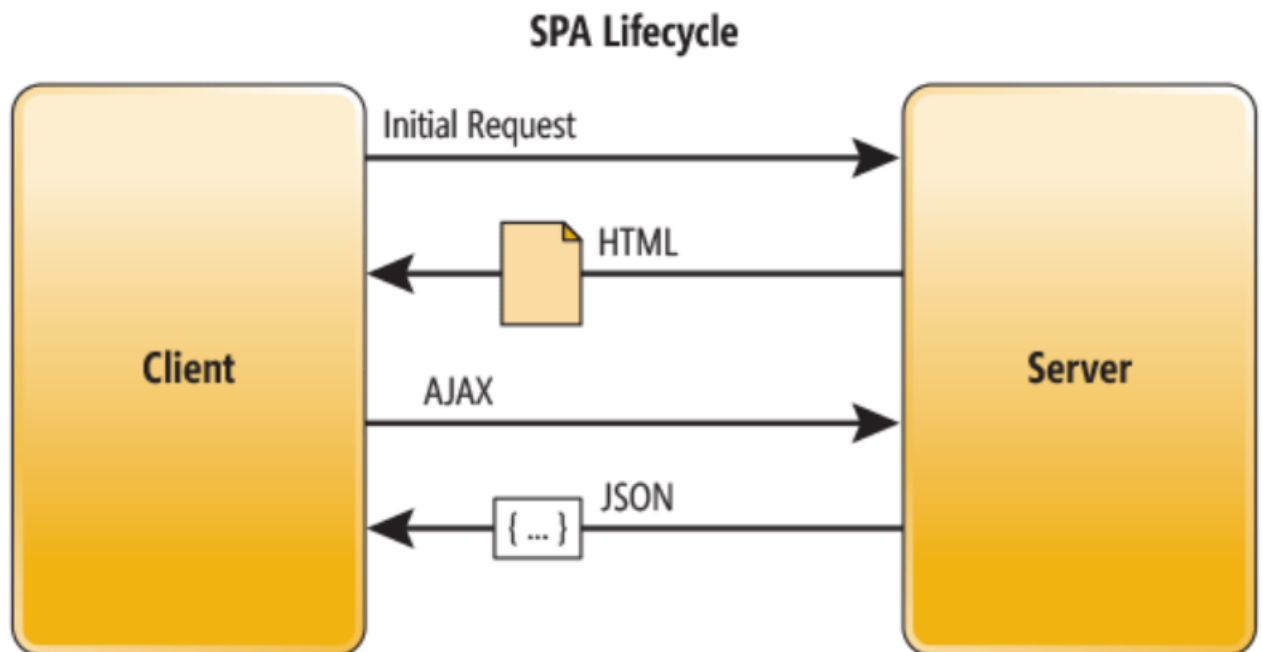


Рисунок 1.6 - Схема SPA

До його переваг належать:

- покращений користувацький досвід: весь код додатку завантажується один раз під час початкового завантаження, а надалі навігація і взаємодія з додатком відбуваються без повного перезавантаження

сторінки. Це призводить до більш плавної і швидкої реакції на дії користувача, що поліпшує загальний користувацький досвід;

- зменшення завантаження сторінок: сторінки не перезавантажуються, тому немає необхідності завантажувати повторно всі ресурси, такі як HTML, CSS, JavaScript, з кожною навігацією. Це зменшує час завантаження сторінок і покращує продуктивність додатка;
- зменшення серверного навантаження: частину обробки відображення інтерфейсу користувача відбувається на клієнтському боці. Це означає, що сервер несе менше навантаження, оскільки він відповідає головним чином за надання даних, а не за рендеринг сторінок;
- більш гнучка розробка: забезпечує більш гнучку розробку, оскільки фронтенд і бекенд можуть працювати незалежно один від одного. Це дозволяє командам розробників працювати паралельно та використовувати різні технології для фронтенду і бекенду.

До недоліків належать:

- більш навантажена клієнтська сторона: весь код додатку завантажується на клієнтському пристрої, що може вимагати більше ресурсів (пам'яті, процесора) для виконання;
- SEO-оптимізація: Традиційні пошукові системи не завжди ефективно індексують вміст, який генерується динамічно на клієнтському боці.

Проаналізувавши вищеперелічені архітектури, було обрано використовувати схему SPA для забезпечення користувачу досвіду, близького до користування настільною програмою.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Розробка обраного програмного продукту передбачає використання певних інструментальних засобів. Для написання бекенд частини додатку було

вирішено використовувати редактор PhpStorm, а фронтенд частини – WebStorm, через наявність студентської підписки, що надається компанією JetBrains.

Безпосередньо для розробки бекенд частини було обрано мову програмування PHP. Для неї існує велика кількість фреймворків, але найпопулярніші з них – це Symfony [13], Laravel та Yii2 [14]. Вже з вказаних фреймворків було обрано Laravel, бо він, на відміну від Yii2, дозволяє створювати API без витрачання додаткових ресурсів та, на відміну від Symfony, має велику спільноту та гарну документацію.

Для клієнтської частини буде використовуватися бібліотека мови програмування JavaScript – React. На даний момент, це найпопулярніше рішення для створення додатків згідно архітектури SPA. Якщо порівнювати з конкурентами, Vue [15] та Angular [16], то React має одну з найбільших спільнот розробників, що призводить до ширшої підтримки, більшої кількості розширень та доступу до багатofункціональних бібліотек.

В якості системи керування базами даних обрано MySQL. Вибір було зупинено на даній технології через наступні її переваги:

- широка підтримка: має велику спільноту розробників і користувачів, що надає підтримку, документацію та різноманітні ресурси для вирішення проблем і отримання допомоги;
- висока продуктивність: відома своєю високою продуктивністю і швидкодією. Вона добре масштабується для обробки великих обсягів даних та високого навантаження;
- легка інтеграція: легко інтегрується з іншими технологіями та мовами програмування, що дозволяє використовувати його в різноманітних додатках та середовищах;
- безпека: надає різні механізми безпеки, такі як рівні доступу, шифрування даних, механізми аутентифікації та контролю доступу до бази даних.

В якості локального вебсервера буде використовуватися OpenServer. Він дозволяє розробникам запускати та тестувати вебсайти або вебдодатки на своєму

комп'ютері без необхідності публікації на віддаленому сервері. Він надає вбудований набір програмного забезпечення, такий як вебсервер Apache, СКБД MySQL, PHP, phpMyAdmin та інші інструменти.

Для забезпечення ефективної роботи системи, максимальної продуктивності та оптимального використання апаратних ресурсів, були встановлені наступні вимоги:

1. Вимоги до серверного обладнання:

- оперативна пам'ять – 8 GB;
- жорсткий диск - 256 GB;
- операційна система – Linux;
- процесор з частотою від 3 GHz;
- node.js версії 18 і вище.

2. Вимоги до робочої станції користувача:

- підтримка браузера *Google Chrome* версії 45+ або;
- підтримка браузера *Mozilla Firefox* версії 38+ або;
- підтримка браузера *Microsoft Edge* версії 12+ або;
- підтримка браузера *Opera* версії 30+ або;
- підтримка браузера *Safari* версії 9+.

Висновки до першого розділу

У даному розділі було виконано постановку задачі на кваліфікаційну роботу. Визначено функціональні вимоги та основні етапи роботи.

Проведено аналіз аналогів до продукту, що розроблюється. І завдяки цьому отримано наступні критерії для розробки: інтуїтивний та зрозумілий дизайн; наявність відеоконтенту; наявність коментарів; українська мова інтерфейсу; достатня кількість курсів; розподіл за категоріями; наявність вправ для перевірки своїх знань.

Було проаналізовано існуючі архітектури і обрано використовувати схему SPA для забезпечення користувачу досвіду, близького до користування настільною програмою.

Також, було обґрунтовано вибір використання наступного стеку технологій для реалізації застосунку: Laravel для побудови бекенд частини, React – для фронтенд частини; в якості системи керування базами даних обрано MySQL; в якості локального вебсервера буде використовуватися OpenServer.

Були встановлені вимоги до серверного обладнання та робочої станції користувача.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ОНЛАЙН-СЕРВІСУ ПРОХОДЖЕННЯ КУРСІВ УКРАЇНСЬКОЇ МОВИ

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

Для наочного зображення відношення між акторами та прецедентами в системі, було побудовано діаграму варіантів використання [17].

На рис. 2.1 зображено побудовану діаграму варіантів використання.

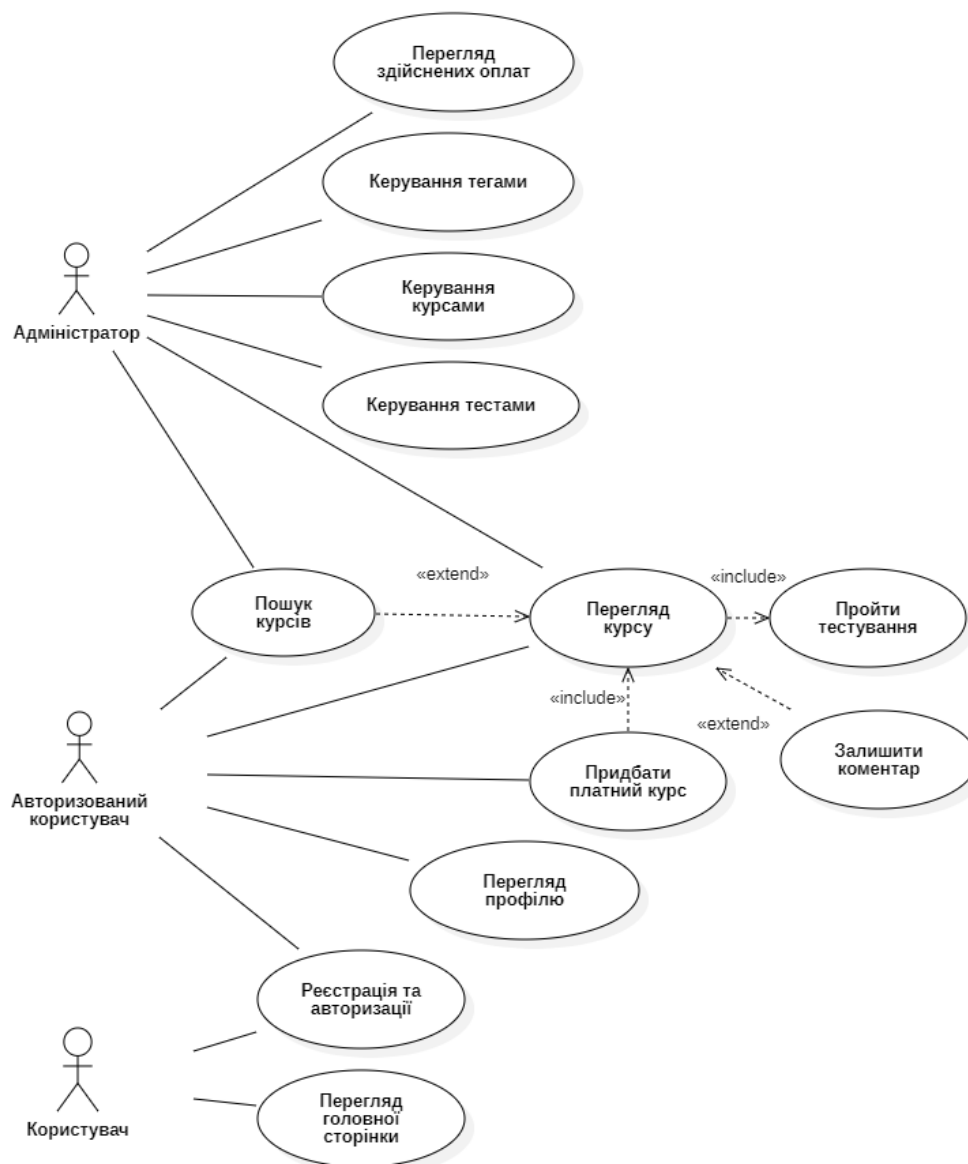


Рисунок 2.1 - Діаграма варіантів використання

Відповідно до побудованої діаграми можна виокремити наступні вимоги системи.

Бізнес-вимоги:

1. Основні цілі: проект створюється з метою вивчення інформації за допомогою онлайн-ресурсу.
2. Можливості: перегляд курсів та придбання платних курсів.
3. Представлення проекту: проект буде реалізовано у вигляді веб-додатку, що містить зручний інтерфейс та необхідну інформацію за даною тематикою.

Вимоги користувачів:

- Зовнішні користувачі – авторизований користувач:
 1. Можливість пошуку курсу.
 2. Можливість фільтрації та сортування курсів.
 3. Можливість переглядати обраний курс та залишити коментар про нього.
 4. Можливість пройти тестування за тематикою курсу.
 5. Можливість придбати платний курс онлайн і відразу мати до нього доступ.
 6. Можливість перегляду профілю та розпочатих курсів.
- Внутрішній користувач – адміністратор:
 1. Керування вмістом вебдодатку.
 2. Перегляд успішних оплат курсів.

Функціональні вимоги:

1. Підключення сторонніх API для реалізації оплати і авторизації за допомогою Google.
2. Створення функціоналу для проходження тестів.
3. Підключення пакетів для пришвидшення процесу розробки.

Нефункціональні вимоги:

1. Сприйняття:
 - час, потрібний для навчання інструментами роботи з інформаційною системою для звичайних користувачів – 2-3 години, для досвідчених – 1 година;

- час відповіді системи для звичайних запитів не повинен перевищувати 5 секунд, а для більш складних запитів – 10 сек;
- інтерфейс повинен бути інтуїтивно зручним для користувача та не вимагати додаткової підготовки.

2. Продуктивність

- система повинна підтримувати мінімум 50 користувачів, що одночасно працюють, пов'язаних з спільною базою даних.

Бізнес правила:

Система повинна відповідати всім стандартам побудови сучасних веб-застосунків, що використовує архітектуру SPA (Single-page Application).

Антивимоги:

- не використовує інформацію, заборонену нормами моралі та законодавством;
- не використовує інформацію, що є недійсною або може зашкодити психіці користувачів;
- не містить реклами.

Бізнес-логіка:

Система пропонує як безкоштовні курси, так і платні, що можна придбати на сайті та відразу їх проходити.

Зважаючи на те, що бекенд частина є об'єктно-орієнтованою, побудуємо діаграму класів [18] репозиторіїв вебдодатку.

На рис. 2.2 зображено діаграму класів репозиторіїв додатку.

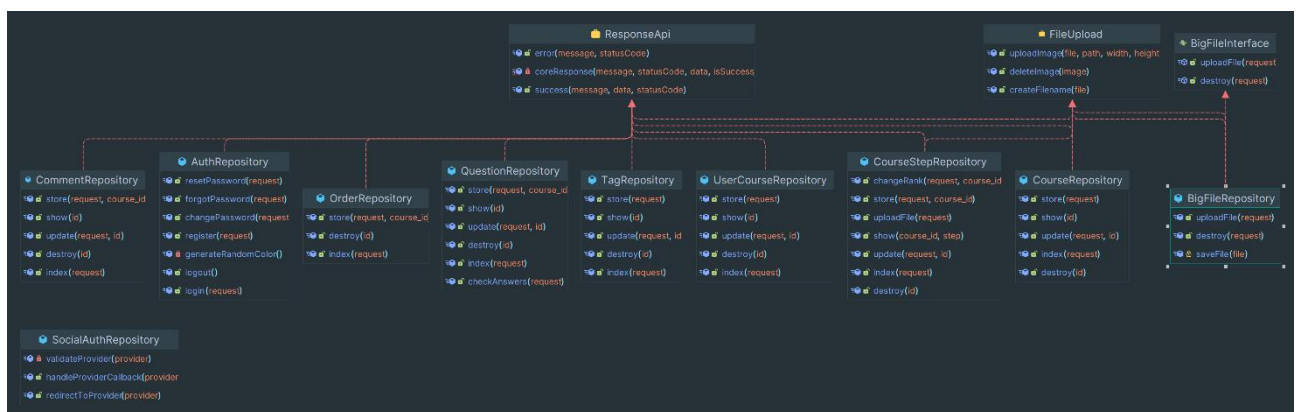


Рисунок 2.2 - Діаграма класів репозиторіїв додатку

Нині ІТ-сфера розвивається дуже швидко, тому рішення багатьох проблем, що часто зустрічаються, вже знайдено і надано назву “патерн проектування”. Без їх використання важко уявити сучасні застосунки. Тому використання патернів продумується ще на стадії моделювання майбутньої системи.

Отже, було прийнято рішення використовувати патерн “Репозиторій” [19], який напевно є найпопулярнішим рішенням для застосунків, розроблених за архітектурою SPA.

“Репозиторій” – це класи або компоненти, які інкапсулюють логіку, необхідну для доступу до джерел даних. Вони централізують спільну функціональність доступу до даних, забезпечуючи кращу зручність обслуговування та відокремлюючи інфраструктуру або технологію, що використовується для доступу до баз даних, від рівня моделі предметної області. Часто використовується із патерном “Unit Of Work”.

На рис. 2.3 зображено реалізацію патерну “Репозиторій” для блоку авторизації.

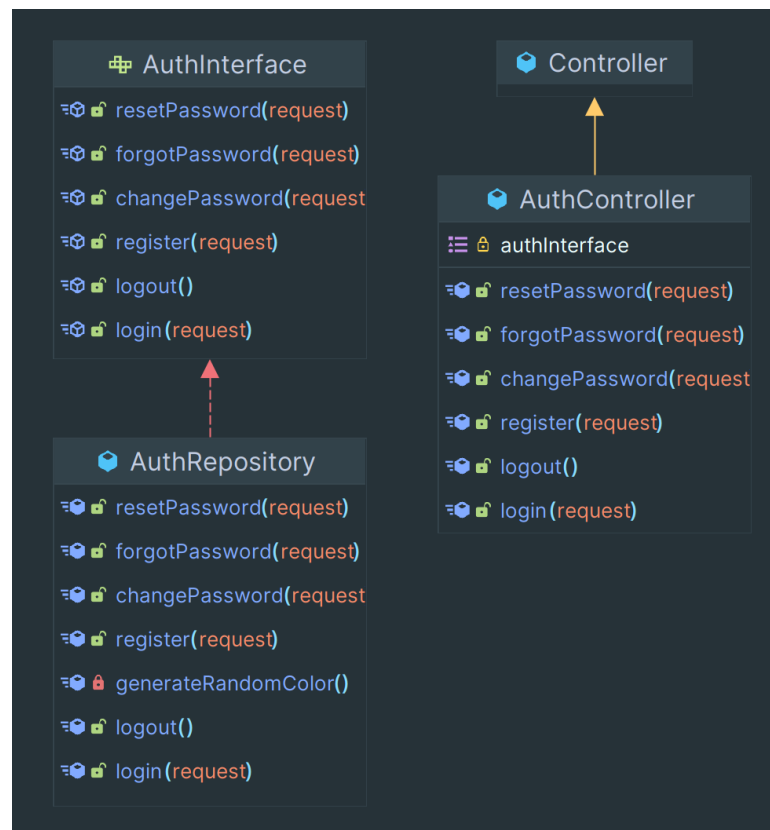


Рисунок 2.3 - Реалізація патерну “Репозиторій” для блоку авторизації

На рис. 2.4 зображено загальний вигляд патерну “Репозиторій”.

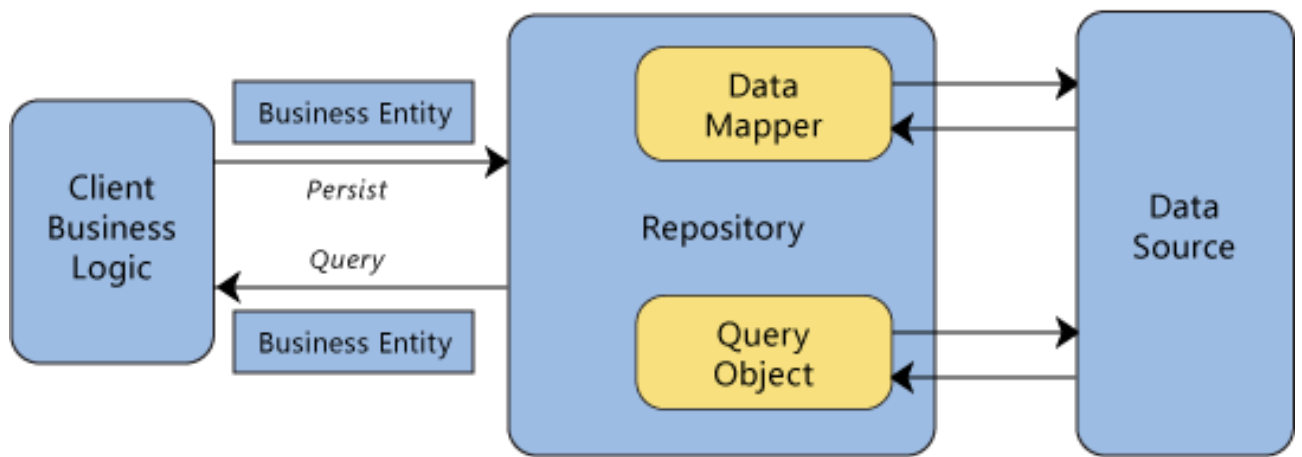


Рисунок 2.4 - Загальний вигляд патерну “Репозиторій”

Також, для побудови системи буде використовуватися патерн “Unit Of Work”. Проте потреби його реалізовувати немає, бо більшість фреймворків надають цей патерн з коробки.

Це патерн об’єктно-реляційної поведінки, мета якого полягає у відстежуванні зміни об’єктів під час транзакції. Під час роботи із базою даних важливо відстежувати зміни в об’єктах, в іншому випадку дані не будуть оновлені. Це також правильно для операцій додавання та видалення.

Можна змінювати дані в сховищі при кожній взаємодії з об’єктом, але це призведе до багатьох викликів до бази даних. Очевидно, це потребує підтримування транзакції відкритою, що впливає на продуктивність роботи.

Даний шаблон пропонує відстежувати всі зміни над об’єктами та вносити їх у вигляді єдиної транзакції.

На рис. 2.5 зображено загальний вигляд патерну “Unit Of Work”.

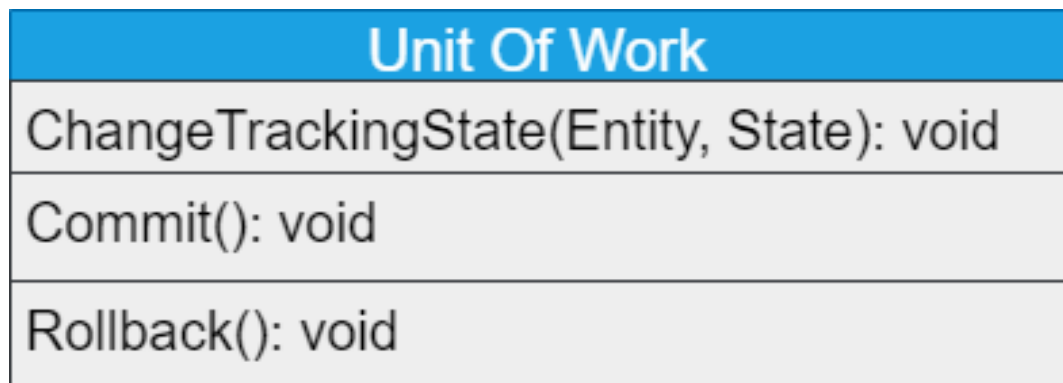


Рисунок 2.5 - Загальний вигляд патерну “Unit Of Work”

У клієнтській частині додатку не передбачено використання класів, оскільки новітня версія бібліотеки React перейшла до принципу функціонального програмування.

2.2. Розробка бази даних системи

Для збереження даних було обрано СКБД MySQL, бо він є одним з найпопулярніших та добре підтримуваних СКБД на сьогоднішній день. MySQL має широкий функціонал, надійність та швидкодію, що робить його чудовим вибором для збереження даних. Крім того, MySQL є безкоштовним та відкритим, що дає можливість економити на ліцензійних витратах.

На рис. 2.6 продемонстровано ER-діаграму спроектованої бази даних.

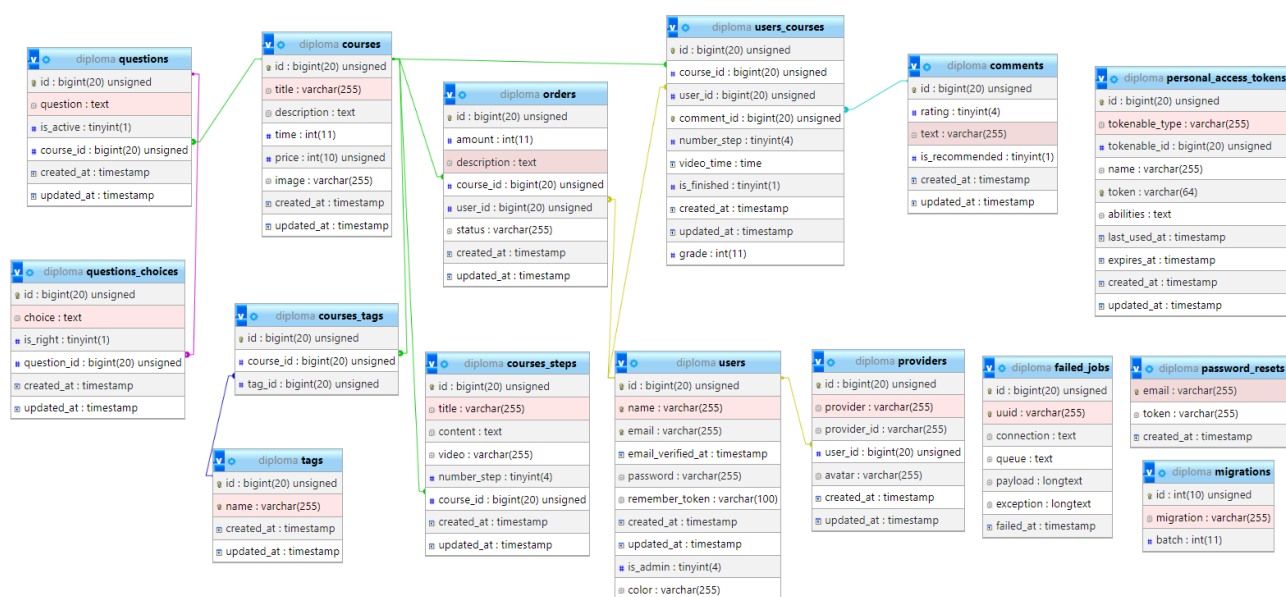


Рисунок 2.6 - ER-діаграма спроектованої бази даних

Опис таблиць, полів та їх призначення наведені в таблицях 2.1-2.15.

Таблиця 2.1

Структура таблиці “users”

Поле	Ключ	Тип	Призначення
1.	2.	3.	4.
id	РК	bigint	Ідентифікатор
name	-	varchar(255)	Нікнейм користувача
email	-	varchar(255)	Електронна пошта користувача

1.	2.	3.	4.
email_verified_at	-	timestamp	Дата та час верифікації електронної пошти
password	-	varchar(255)	Пароль користувача
remember_token	-	varchar(100)	Токен для запам'ятовування користувача
created_at	-	timestamp	Дата та час створення акаунту
updated_at	-	timestamp	Дата та час оновлення акаунту
is_admin	-	tinyint(4)	Чи є користувач адміністратором
color	-	varchar(255)	Колір аватара користувача

Таблиця 2.2

Структура таблиці “password_resets”

Поле	Ключ	Тип	Призначення
email	РК	varchar(255)	Електронна пошта користувача
token	-	varchar(255)	Токен для скидання пароля
created_at	-	timestamp	Дата та час створення токена скидання

Таблиця 2.3

Структура таблиці “migrations”

Поле	Ключ	Тип	Призначення
id	РК	int	Ідентифікатор
migration	-	varchar(255)	Назва міграції
batch	-	int	Номер партії додавання

Таблиця 2.4

Структура таблиці “failed_jobs”

Поле	Ключ	Тип	Призначення
id	РК	bigint	Ідентифікатор
uuid	-	varchar(255)	Унікальний номер
connection	-	text	З'єднання
queue	-	text	Черга
payload	-	longtext	Корисне навантаження
exception	-	longtext	Виключення
failed_at	-	timestamp	Дата та час виникнення помилки

Таблиця 2.5

Структура таблиці “personal_access_tokens”

Поле	Ключ	Тип	Призначення
id	PK	bigint	Ідентифікатор
tokenable_type	-	varchar(255)	Тип моделі токена
tokenable_id	-	text	Ідентифікатор моделі токена
name	-	text	Назва токена
token	-	longtext	Токен
abilities	-	longtext	Можливості
last_used_at	-	timestamp	Дата та час останнього використання токена
expires_at	-	timestamp	Дата та час закінчення токена
created_at	-	timestamp	Дата та час створення токена
updated_at	-	timestamp	Дата та час оновлення токена

Таблиця 2.6

Структура таблиці “providers”

Поле	Ключ	Тип	Призначення
id	PK	bigint	Ідентифікатор
provider	-	varchar(255)	Назва провайдера
provider_id	-	varchar(255)	Ідентифікатор провайдера
user_id	FK	bigint	Ідентифікатор користувача (зв’язаний з users)
avatar	-	varchar(255)	Аватар користувача
created_at	-	timestamp	Дата та час створення провайдера
updated_at	-	timestamp	Дата та час оновлення провайдера

Таблиця 2.7

Структура таблиці “courses”

Поле	Ключ	Тип	Призначення
1.	2.	3.	4.
id	PK	bigint	Ідентифікатор
title	-	varchar(255)	Назва курсу
description	-	text	Опис курсу
time	-	int	Час проходження

Продовження табл. 2.7

1.	2.	3.	4.
price	-	int	Ціна
image	-	varchar(255)	Назва картинки
created_at	-	timestamp	Дата та час створення курсу
updated_at	-	timestamp	Дата та час оновлення курсу

Таблиця 2.8

Структура таблиці “tags”

Поле	Ключ	Тип	Призначення
id	PK	bigint	Ідентифікатор
name	-	varchar(255)	Назва тегу
created_at	-	timestamp	Дата та час створення тегу
updated_at	-	timestamp	Дата та час оновлення тегу

Таблиця 2.9

Структура таблиці “courses_tags”

Поле	Ключ	Тип	Призначення
id	PK	bigint	Ідентифікатор
course_id	FK	bigint	Ідентифікатор курсу (зв’язаний з courses)
tag_id	FK	bigint	Ідентифікатор тегу (зв’язаний з tags)

Таблиця 2.10

Структура таблиці “questions”

Поле	Ключ	Тип	Призначення
id	PK	bigint	Ідентифікатор
question	-	text	Запитання
is_active	-	tinyint(1)	Чи активне запитання
course_id	FK	bigint	Ідентифікатор курсу (зв’язаний з courses)
created_at	-	timestamp	Дата та час створення запитання
updated_at	-	timestamp	Дата та час оновлення запитання

Таблиця 2.11

Структура таблиці “questions_choices”

Поле	Ключ	Тип	Призначення
id	PK	bigint	Ідентифікатор
choice	-	text	Варіант відповіді
is_right	-	tinyint(1)	Чи правильний варіант
question_id	FK	bigint	Ідентифікатор запитання (зв’язаний з questions)
created_at	-	timestamp	Дата та час створення варіанту
updated_at	-	timestamp	Дата та час оновлення варіанту

Таблиця 2.12

Структура таблиці “orders”

Поле	Ключ	Тип	Призначення
id	PK	bigint	Ідентифікатор
amount	-	int	Вартість замовлення
description	-	text	Опис замовлення
course_id	FK	bigint	Ідентифікатор курсу (зв’язаний з courses)
user_id	FK	bigint	Ідентифікатор користувача (зв’язаний з users)
status	-	varchar(255)	Статус замовлення
created_at	-	timestamp	Дата та час створення замовлення
updated_at	-	timestamp	Дата та час оновлення замовлення

Таблиця 2.13

Структура таблиці “courses_steps”

Поле	Ключ	Тип	Призначення
1.	2.	3.	4.
id	PK	bigint	Ідентифікатор
title	-	varchar(255)	Назва розділу
content	-	text	Контент розділу
video	-	varchar(255)	Назва відео

Продовження табл. 2.13

1.	2.	3.	4.
number_step	-	tinyint(4)	Крок розділу
course_id	FK	bigint	Ідентифікатор курсу (зв'язаний з courses)
created_at	-	timestamp	Дата та час створення розділу
updated_at	-	timestamp	Дата та час оновлення розділу

Таблиця 2.14

Структура таблиці “comments”

Поле	Ключ	Тип	Призначення
id	PK	bigint	Ідентифікатор
rating	-	tinyint(4)	Оцінка курсу
text	-	varchar(255)	Текст коментаря
is_recommended	-	varchar(255)	Чи рекомендує
created_at	-	timestamp	Дата та час створення коментаря
updated_at	-	timestamp	Дата та час оновлення коментаря

Таблиця 2.15

Структура таблиці “users_courses”

Поле	Ключ	Тип	Призначення
id	PK	bigint	Ідентифікатор
course_id	FK	bigint	Ідентифікатор курсу (зв'язаний з courses)
user_id	FK	bigint	Ідентифікатор користувача (зв'язаний з users)
comment_id	FK	bigint	Ідентифікатор коментаря (зв'язаний з comments)
number_step	-	tinyint(4)	Поточний крок розділу
video_time	-	time	Поточний час відео
is_finished	-	tinyint(1)	Чи завершений
grade	-	int	Оцінка
created_at	-	timestamp	Дата та час створення замовлення
updated_at	-	timestamp	Дата та час оновлення замовлення

2.3. Проектування та реалізація алгоритмів роботи системи

Так як обраної архітектурою програмного додатку є SPA, то вебзастосунок має явне розподілення на серверну та клієнтську частину і передбачає використання підходу RESTful API [20].

RESTful API (Representational State Transfer API) є набором правил та протоколів, які дозволяють різним програмним додаткам взаємодіяти між собою через мережу, зазвичай через Інтернет.

Можна виділити наступні основні характеристики та принципи RESTful API:

- ресурси: побудований навколо ресурсів, які зазвичай представляються у вигляді URL-адрес. Кожен ресурс має унікальний URL, а клієнти взаємодіють з цими ресурсами, виконуючи HTTP-запити до відповідних URL-адрес;
- методи HTTP: RESTful API використовує стандартні методи HTTP для виконання різних дій з ресурсами. Методи, що використовується під час розробки додатку:
 1. GET: Отримує представлення ресурсу.
 2. POST: Створює новий ресурс.
 3. PUT: Оновлює існуючий ресурс.
 4. PATCH: Оновлює частину існуючого ресурсу.
 5. DELETE: Видаляє ресурс.

У випадку з додатком, що створюється відбувається наступне. З клієнтської частину відбуватиметься звернення до приватного API для отримання необхідних даних. Для цього за допомогою фреймворку Laravel повертається JSON-об'єкт при зверненні до відповідного ендпоінту з клієнтської частини за допомогою бібліотеки Axios. На стороні серверу відбувається отримання даних з бази даних і здійснення маніпуляцій на ними. Після чого ці дані використовуються для генерації користувацького інтерфейсу. Якщо ж

користувач змінює запит сторінки, то повторно відбувається запит до серверу вебзастосунку.

На рис. 2.7 зображено загальний алгоритм роботи додатку.

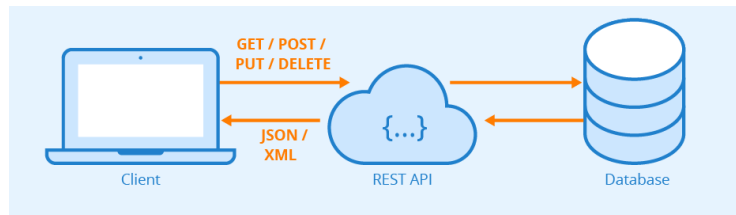


Рисунок 2.7 - Загальний алгоритм роботи додатку

Також, для зображення алгоритму роботи додатку побудуємо діаграму активностей [21].

На рис. 2.8 зображено діаграму активностей.

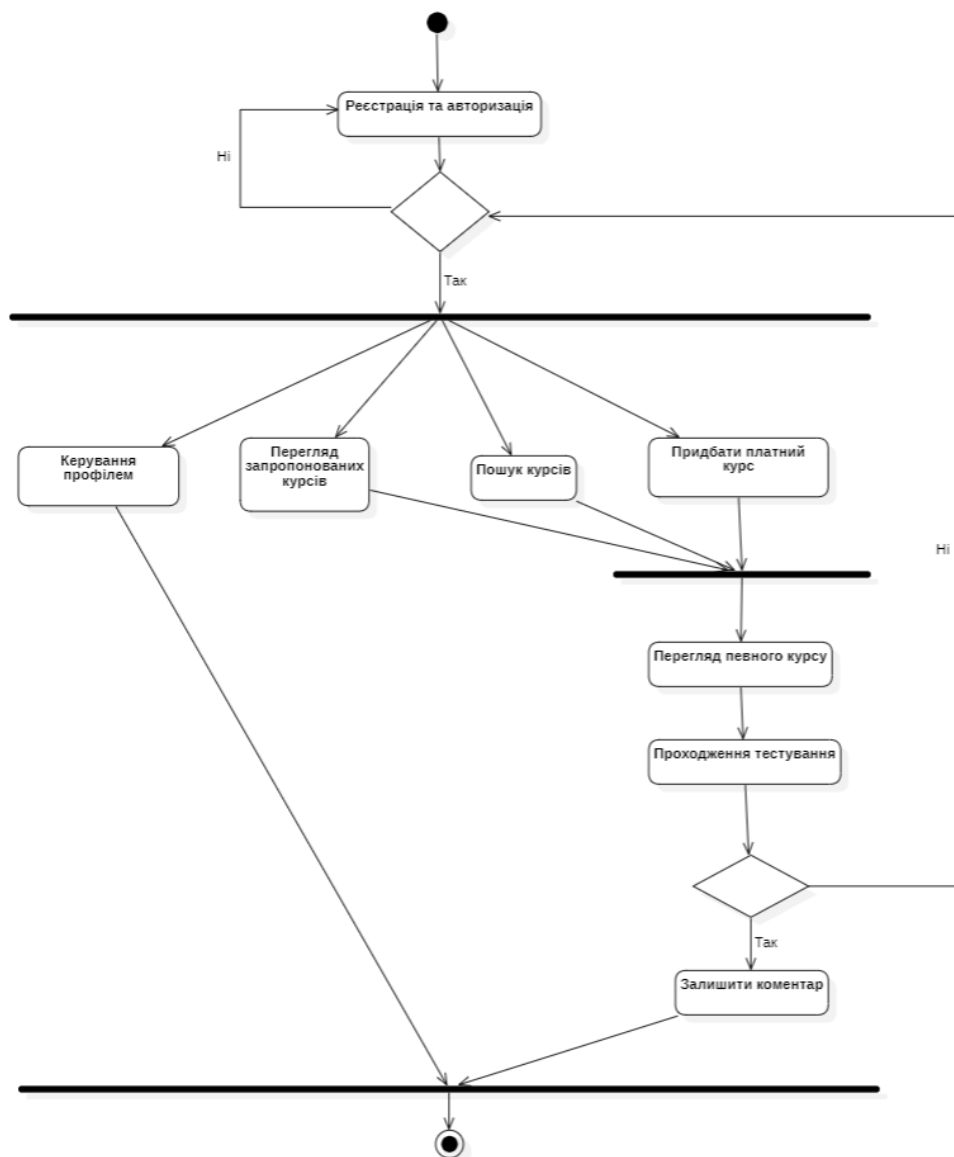


Рисунок 2.8 – Діаграма активностей системи

Опис отриманої діаграми:

1. Авторизація (так – перехід до використання системи, ні – перехід на початок з повідомленням про необхідність наявності акаунту).
2. Керування профілем.
3. Перегляд запропонованих курсів, пошук курсів, придбати платний курс:
 - перегляд певного курсу;
 - проходження тестування (так – перехід до створення коментаря, ні – перехід до використання системи або кінець роботи додатку);
 - залишити коментар.
4. Кінець роботи веб-застосунку.

Безпосередньо застосунок можна умовно поділити на певні модулі. Для презентативності побудуємо діаграму компонентів додатку.

На рис. 2.9 зображено діаграму компонентів [22] додатку.

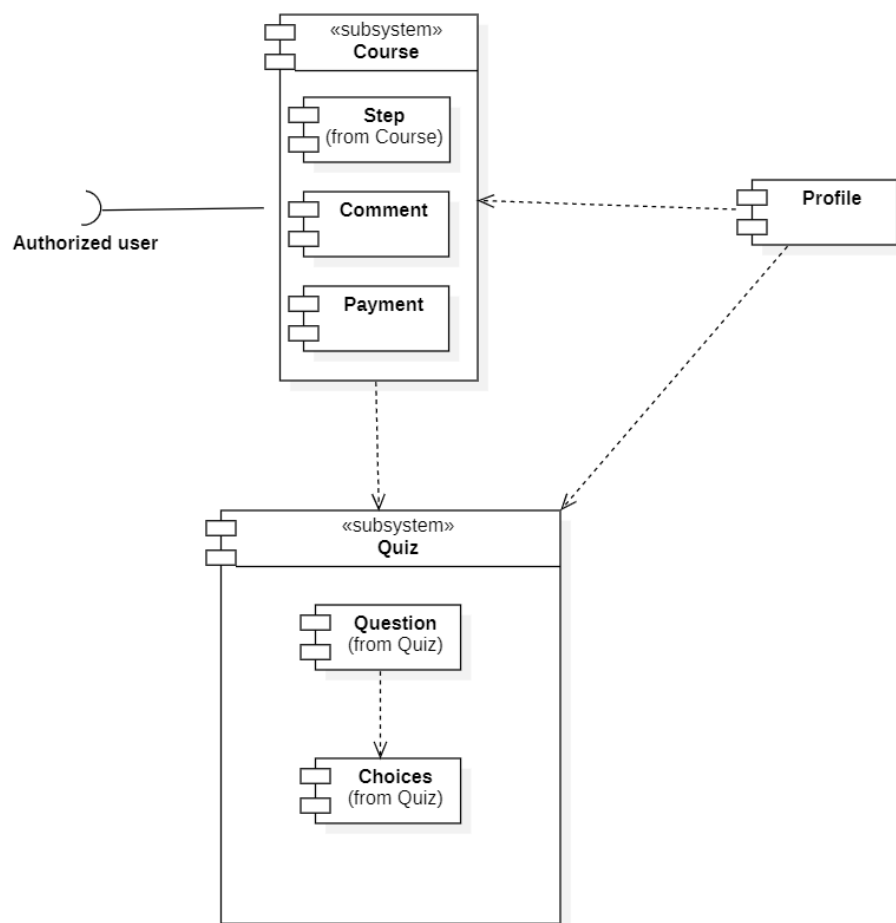


Рисунок 2.9 - Діаграма компонентів додатку

Проаналізувавши отриману діаграму, отримуємо наступне:

1. Для роботи з основними компонентами користувач повинен бути авторизованим у системі.
2. Систему було розділено на два підкомпоненти і один окремий компонент.
3. Підсистема Course представляє собою проходження певного курсу, що поділений на кроки, його оплати та залишення коментаря до нього.
4. Підкомпонент Quiz відповідає за тестування, що є обов'язковим критерієм закінчення відповідного курсу.
5. Компонент Profile відповідає за керування профілем та відстежування прогресу користувачем.

Детальніше розглянемо алгоритм оплати. Для цього було побудовано діаграму послідовності.

На рис. 2.10 зображено діаграму послідовності [23].

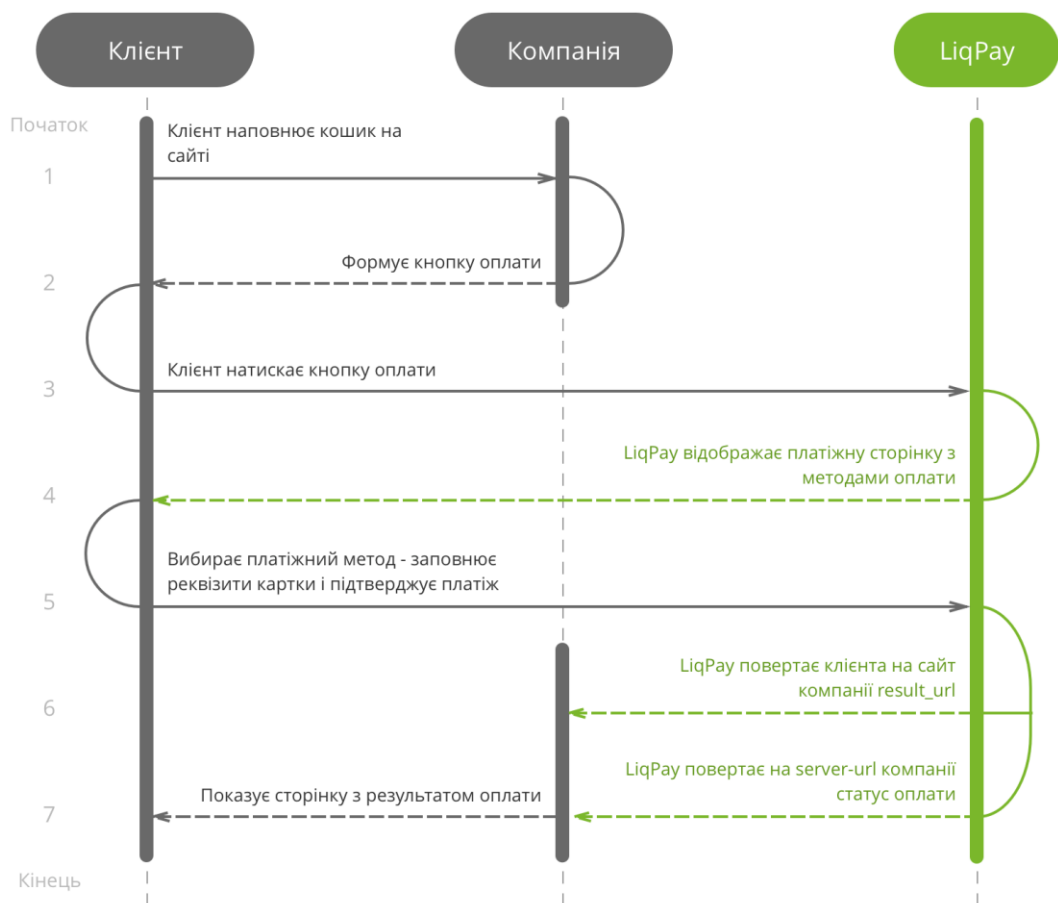


Рисунок 2.10 - Діаграма послідовності для оплати

У разі якщо обраний курс є платним, то користувач повинен спочатку його купити. На сторінці курсу користувач натискає відповідну кнопку, при цьому створюється нове замовлення у системі. При підтвердженні оформлення замовлення користувача перенаправляє на платіжну сторінку, інакше замовлення видаляється з системи. Після успішної оплати оновлюється статус замовлення і у користувача з'являється доступ до курсу.

2.4. Реалізація онлайн-сервісу проходження курсів української мови

Будь-який початок роботи з додатком передбачає авторизацію користувача. Тому розглянемо реалізацію даного функціоналу.

На рис. 2.11 зображено метод login класу AuthRepository.

```
public function login(Request $request): JsonResponse
{
    $login = $request->login;
    $fieldType = filter_var($login, filter: FILTER_VALIDATE_EMAIL) ? 'email' : 'name';
    request()->merge([$fieldType => $login]);
    try {
        if (!Auth::attempt($request->only([$fieldType, 'password']))) {
            return $this->error([
                message: 'Не знайдено користувача з вказаними даними.',
                statusCode: Response::HTTP_UNAUTHORIZED);
        }

        $user = auth()->user();

        if ($user->isAdmin()) {
            $token = $user->createToken( name: $user->name . '_AdminToken', ['server:admin']->plainTextToken);
            $user->role = 1;
        } else {
            $token = $user->createToken( name: $user->name . '_Token', [''])->plainTextToken;
            $user->role = 0;
        }

        return $this->success( message: 'Ви успішно увійшли в систему', ['token' => $token, 'user' => $user]);
    } catch (\Throwable $th) {
        return $this->error($th->getMessage());
    }
}
```

Рисунок 2.11 - Метод login класу AuthRepository

Даний метод передбачає визначення даних за якими користувач хоче увійти. Потім відбувається спроба згідно цих даних увійти в систему, створюючи новий токен з відповідними правами, який в подальшому використовуватиметься на клієнтській частині для формування авторизаційного заголовку. У разі введення неправильних даних користувачі повертається повідомлення про помилку.

На рис. 2.12 зображено метод resetPassword класу AuthRepository.

```
public function resetPassword(ResetPasswordRequest $request): JsonResponse
{
    $user = User::where('email', $request->email)->first();

    if (!$user) {
        return $this->error( message: 'Не існує користувача з вказаним email', statusCode: Response::HTTP_UNAUTHORIZED);
    }

    $resetRequest = PasswordReset::where('email', $user->email)->first();

    if (!$resetRequest || $resetRequest->token != $request->token) {
        return $this->error( message: 'Не дійсний токен скидання', statusCode: Response::HTTP_BAD_REQUEST);
    }

    try {
        $user->fill([
            'password' => Hash::make($request->password)
        ]);
        $user->save();
        $user->tokens()->delete();
        $resetRequest->delete();
        return $this->success( message: "Пароль успішно змінено");
    } catch (\Throwable $th) {
        return $this->error($th->getMessage());
    }
}
```

Рисунок 2.12 - Метод resetPassword класу AuthRepository

Даний метод відповідає за зміну пароля, коли користувач забув пароль і вже перейшов за посиланням для відновлення з пошти. Він перевіряє валідність токена та користувача після чого зберігає новий пароль у системі. Якщо сталася якась помилка під час виконання попередніх дій, користувач отримує про неї повідомлення.

Також, необхідно виокремити реалізацію завантаження відео розділу на сервер. Для цього було використано підхід розділення файлу на шматки (chunk)

і надсилення їх на сервер. Після завантаження всіх шматків вони назад перетворюються в один файл. Реалізацію даного процесу зображено на рис. 2.13, а саме представлено метод `uploadFile` класу `BigFileRepository`.

```
public function uploadFile(Request $request): JsonResponse
{
    $validator = Validator::make($request->all(), [
        'file' => 'required|mimetypes:video/mp4'
    ]);

    if ($validator->fails()) {
        return $this->success( message: 'Невідповідний формат', ['error' => 419]);
    }

    $receiver = new FileReceiver( fileIndexOrFile: 'file', $request, HandlerFactory::classFromRequest($request));

    if ($receiver->isUploaded() == false) {
        throw new UploadMissingFileException();
    }

    $save = $receiver->receive();

    if ($save->isFinished()) {
        return $this->saveFile($save->getFile());
    }

    $handler = $save->handler();
    return $this->success( message: 'Chunk успішно передано', ['done' => $handler->getPercentageDone(), 'status' => true]);
}
```

Рисунок 2.13 - Метод `uploadFile` класу `BigFileRepository`

Цей метод використовує допоміжний метод `saveFile`, реалізацію якого зображено на рис. 2.14.

```
protected function saveFile(UploadedFile $file): JsonResponse
{
    $fileName = $this->createFilename($file);
    $mime = str_replace( search: '/', replace: '-', $file->getMimeType());
    $path = Storage::putFileAs($this->save_path, $file, $fileName);

    $path = preg_replace( pattern: '/public/', replacement: '', $path);
    unlink($file->getPathname());

    return $this->success( message: 'Файл успішно збережено', [
        'complete' => true,
        'path' => @App::make( abstract: 'url')->to('/') . '/storage' . $path,
        'name' => $fileName,
        'mime_type' => $mime
    ]);
}
```

Рисунок 2.14 - Метод `saveFile` класу `BigFileRepository`

Також, важливою частиною застосунку є розділи курсів та їх послідовність. Тому реалізація зміни послідовності розділів для адміністратора є обов’язковою складовою розробки додатку. На рис. 2.15 продемонстровано метод `changeRank` класу `CourseStepRepository`, що за це відповідає.

```
public function changeRank(Request $request, int $course_id): JsonResponse
{
    $steps = [];
    foreach ($request->all() as $item) {
        $steps[] = CourseStep::where(['course_id' => $course_id, 'number_step' => $item['number_step']])->first();
    }
    foreach ($steps as $id => $step) {
        $step->number_step = $id + 1;
        $step->save();
    }

    $steps = CourseStep::where(['course_id' => $course_id])->orderBy('number_step')->get();

    return $this->success( message: 'Порядок успішно змінено', ['steps' => $steps]);
}
```

Рисунок 2.15 - Метод `saveFile` класу `BigFileRepository`

Спочатку ми отримуємо початкове положення курсів і записуємо їх до масиву. Після чого циклічно змінюємо на нові значення.

Невід’ємною частиною додатку є тестування і перевірка правильності вибору користувача. Тому на рис. 2.16 продемонстровано метод `checkAnswers` методу `QuestionRepository`, що відповідає за перевірку відповідей.

```
public function checkAnswers(Request $request): JsonResponse
{
    try {
        if (!$userCourse = UserCourse::where(['course_id' => $request->all()[0]['course_id'], 'user_id' => auth()->id()])->first()) {
            return $this->error( message: 'Спочатку необхідно розпочати курс', statusCode: Response::HTTP_FORBIDDEN);
        }
        $answers = [];
        $right = 0;
        foreach ($request->all() as $id => $item) {
            $choice = QuestionChoice::findOrFail($item['choice']['id']);
            $rightAnswer = QuestionChoice::where(['is_right' => 1, 'question_id' => $item['id']])->first();
            if ($choice->is_right) {
                $answers['res'][$id] = ['res' => true];
                $right++;
            } else {
                $answers['res'][$id] = ['res' => false, 'right' => $rightAnswer->choice];
            }
        }
        $total = round( num: $right / count($request->all()) * 100);
        $answers['total'] = $total;
        if ($userCourse->grade < $total || !$userCourse->grade) {
            $userCourse->grade = $total;
        }
        $userCourse->is_finished = 1;
        $userCourse->save();

        return $this->success( message: 'Відповіді успішно опрацьовано', ['answers' => $answers]);
    } catch (\Throwable $th) {
        return $this->error($th->getMessage());
    }
}
```

Рисунок 2.16 - Метод `checkAnswers` методу `QuestionRepository`

Спочатку отримуємо в масив усі відповіді користувача. Потім циклічно знаходимо варіант відповіді та перевіряємо, чи воно правильно. Якщо так, то збільшуємо кількість правильних відповідей, інакше вказуємо правильну відповідь. Після підрахунку оцінки зберігаємо її, якщо вона краща за попередні спроби або це перша спроба. Після цього курс в системі позначається як пройдений для даного користувача.

Висновки до другого розділу

В даному розділі для наочного зображення відношення між акторами та прецедентами в системі, було побудовано діаграму варіантів використання. Було сформовано вимоги до системи. Було обґрунтовано та описано використання патернів при розробці застосунку. Було спроектовано базу даних та описано поля та їх призначення.

Описано, як було використано архітектуру Single-Page Application та обґрунтовано використання підходу RESTful API. Було продемонстровано загальний алгоритм роботи та побудовані наступні діаграми для презентативності: діаграму активностей системи, діаграму послідовності для оплати та діаграма компонентів системи.

Було продемонстровано реалізацію програмного комплексу та наведено фрагменти коду, що відповідають за авторизацію, оплату та завантаження відео на сервер з поясненням принципу їх роботи.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН-СЕРВІСОМ ПРОХОДЖЕННЯ КУРСІВ УКРАЇНСЬКОЇ МОВИ

3.1. Порядок встановлення та налаштування параметрів системи

Для зображення розгортання системи існує однойменна діаграма – діаграма розгортання [24].

Діаграма розгортання – це діаграма, яка показує конфігурацію вузлів обробки часу виконання та компонентів, які на них живуть. Діаграми розгортання – це різновид структурної діаграми, яка використовується для моделювання фізичних аспектів об'єктно-орієнтованої системи. Вони часто використовуються для моделювання статичного вигляду розгортання системи (топології апаратного забезпечення).

На рис. 3.1 зображено діаграму розгортання.

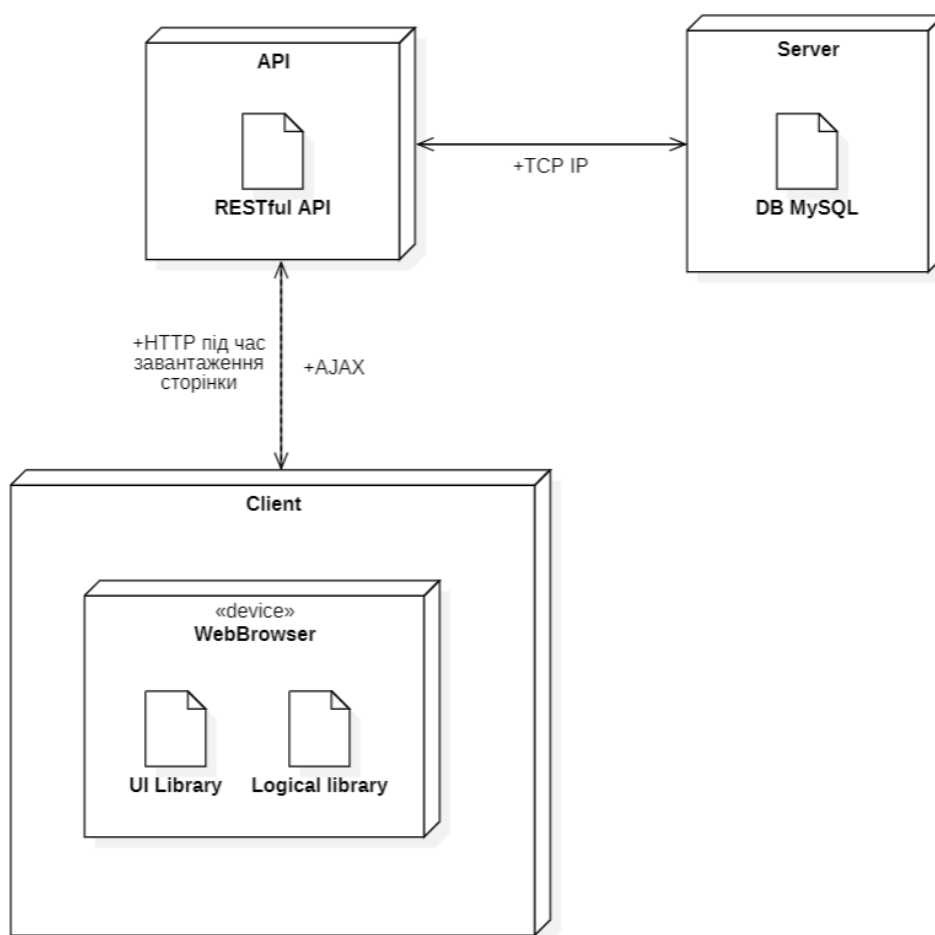


Рисунок 3.1 - Діаграма розгортання додатку

Після побудови даної діаграми можна зробити наступні висновки щодо порядку встановлення та налаштування параметрів системи:

- API вебзастосунку, тобто його бекенд частина, розгортається на порті 8000, в той час як клієнт – на порті 3000;
- бекенд частина звертається до сервера для отримання необхідних ресурсів;
- клієнт представляє собою додаток, що використовує бібліотеки для побудови користувацького інтерфейсу та налаштування логіки оновлення даних і безпосередньо сторінок застосунку;
- клієнт отримає початкову HTML-розмітку під час запуску додатку і в подальшому оновлює її за допомогою асинхронних звернень до API.

3.2. Структура інтерфейсу та порядок взаємодії з онлайн-сервісом

Після відкриття вебзастосунку користувача зустрічає головна сторінка. На рис. 3.2 зображено головну сторінку вебзастосунку.

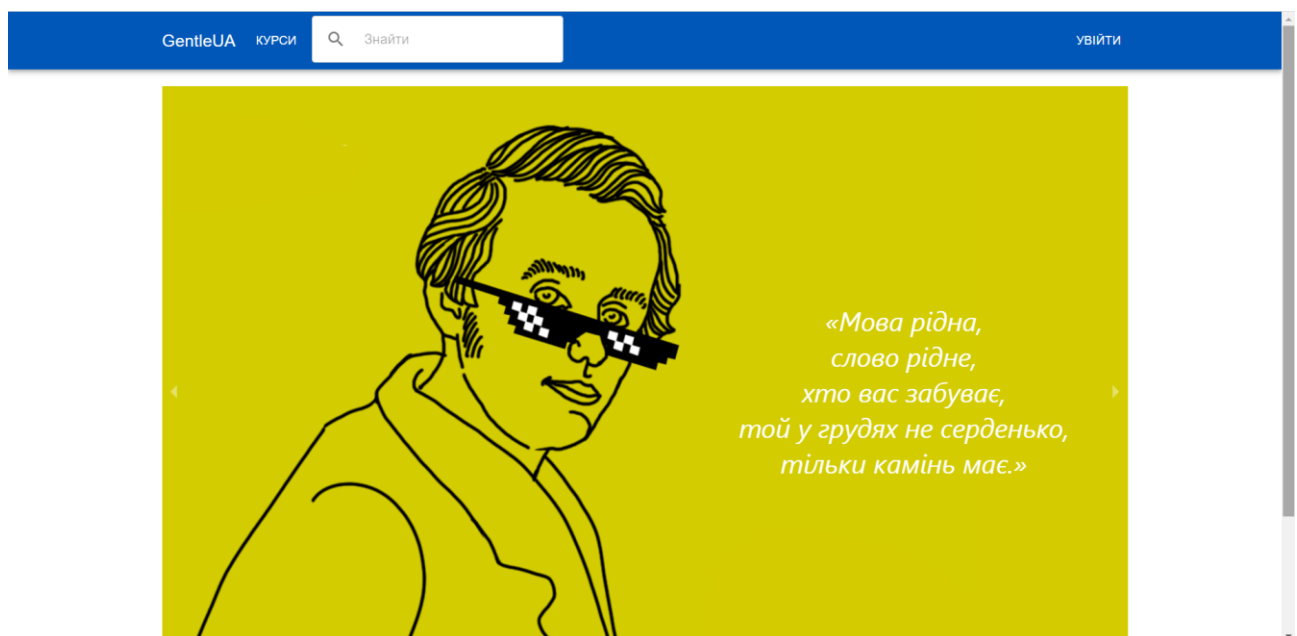


Рисунок 3.2 – Головна сторінка вебзастосунку

Тут він повинен перейти до сторінки входу, аби авторизуватися у системі. У разі якщо користувач ще не має акаунту він може його створити.

На рис. 3.3 зображено сторінку авторизації.

Рисунок 3.3 - Сторінка авторизації

Після успішного входу в систему користувач отримує доступ до переліку всіх курсів.

На рис. 3.4 зображено сторінку курсів.

Рисунок 3.4 - Сторінка курсів

Також, користувач може шукати певний курс за назвою, ввівши пошуковий запит у відповідне поле. Пошук відбувається асинхронно, тобто на кожний введений символ відбувається пошук курсу. На рис. 3.5 зображено даний процес.

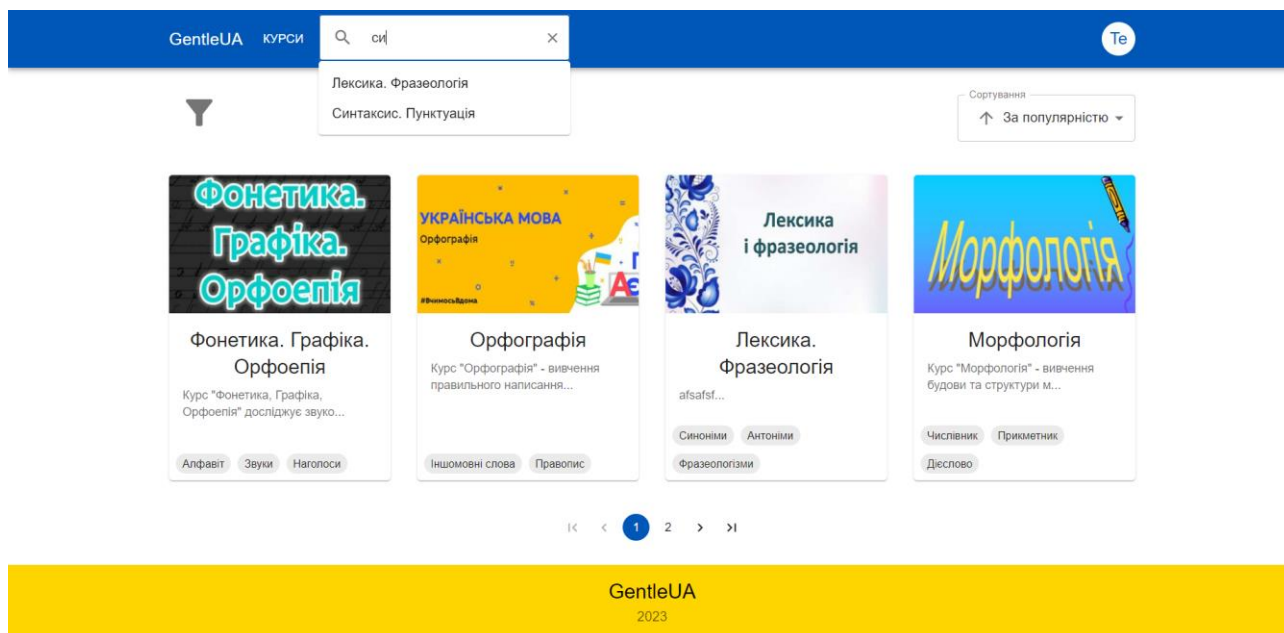


Рисунок 3.5 - Пошук курсів

В системі передбачено сортування і фільтрація курсів. Сортувати можна за наступними критеріями: за популярністю, за датою створення і тривалістю. Фільтрація передбачена за тегами курсів.

На рис. 3.6 зображено відсортовані курси за тривалістю.

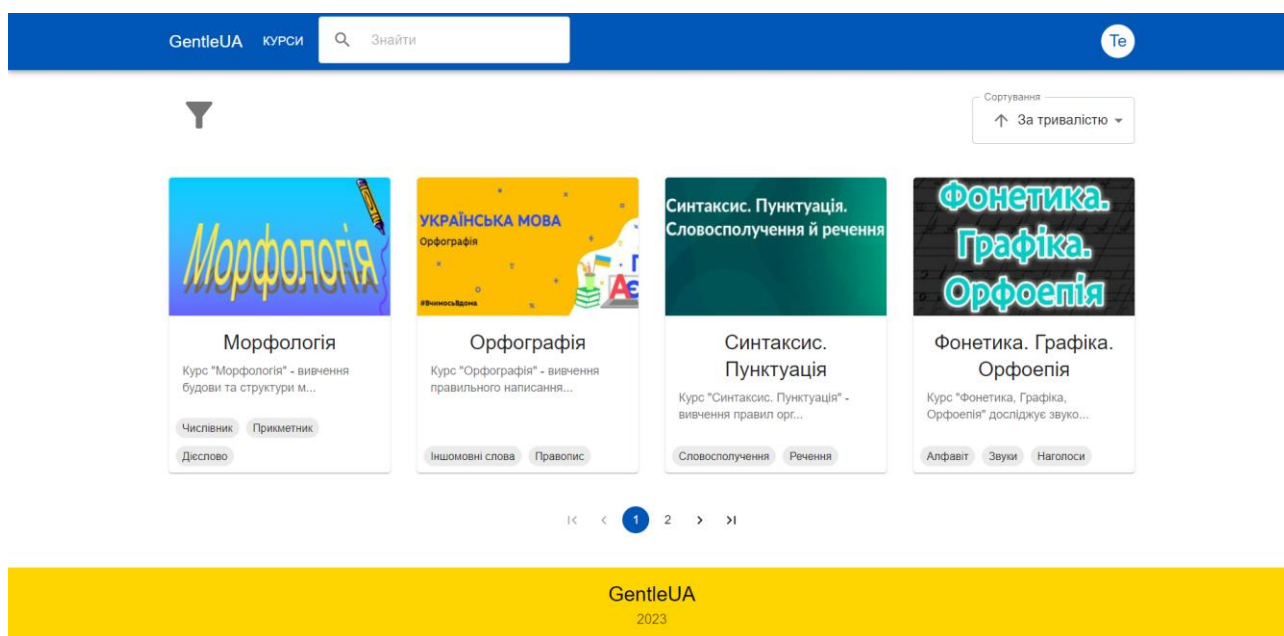


Рисунок 3.6 - Відсортовані курси за тривалістю

Після того, як користувач визначився з курсом він переходить на його сторінку.

На рис. 3.7-3.8 зображено сторінку курсу “Фонетика”.

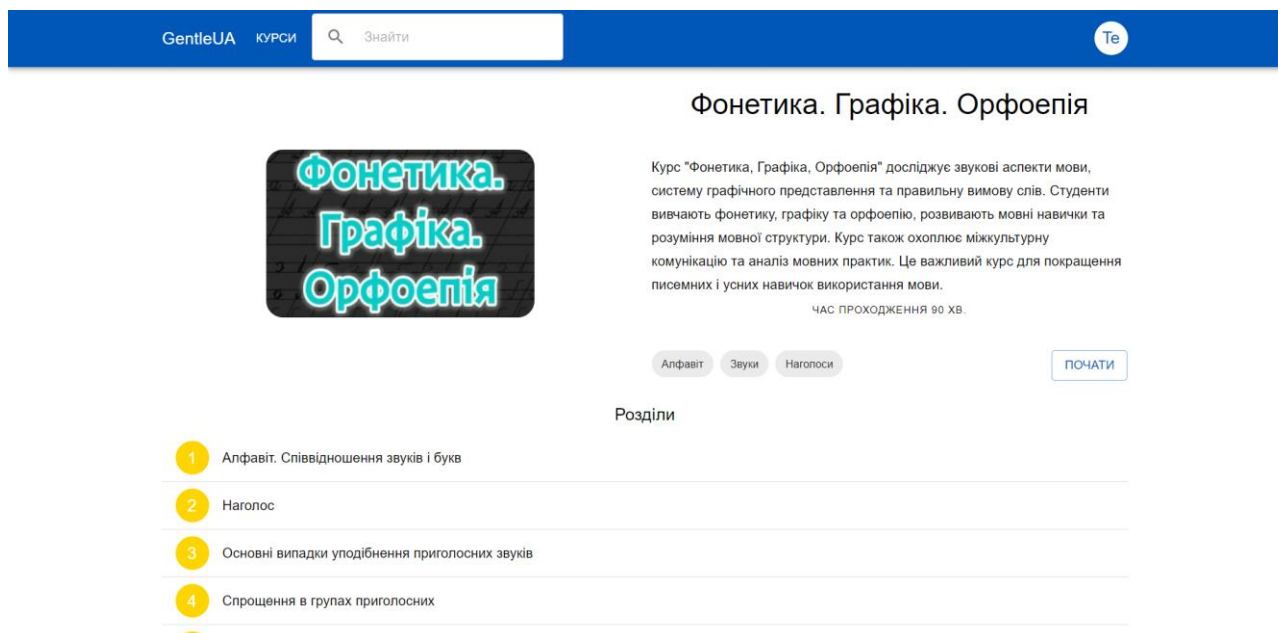


Рисунок 3.7 - Сторінка курсу “Фонетика”

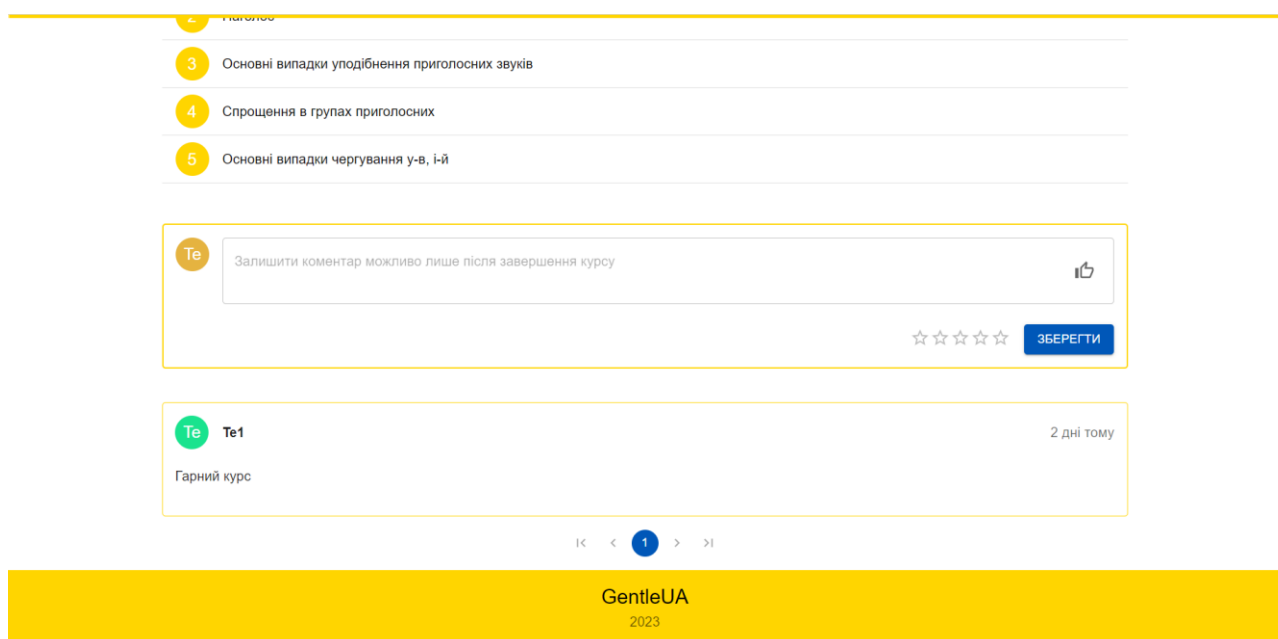


Рисунок 3.8 - Сторінка курсу “Фонетика” (продовження)

Як видно на рисунках вище, користувач може залишити коментар, але тільки після завершення курсу.

Після натискання кнопки “Почати” користувач переходить на перший розділ обраного курсу. На цій сторінці користувач може переглянути відеоконтент і прочитати опорний конспект до розділу.

На рис. 3.9-3.10 зображено розділ “Алфавіт. Співвідношення звуків і букв”.

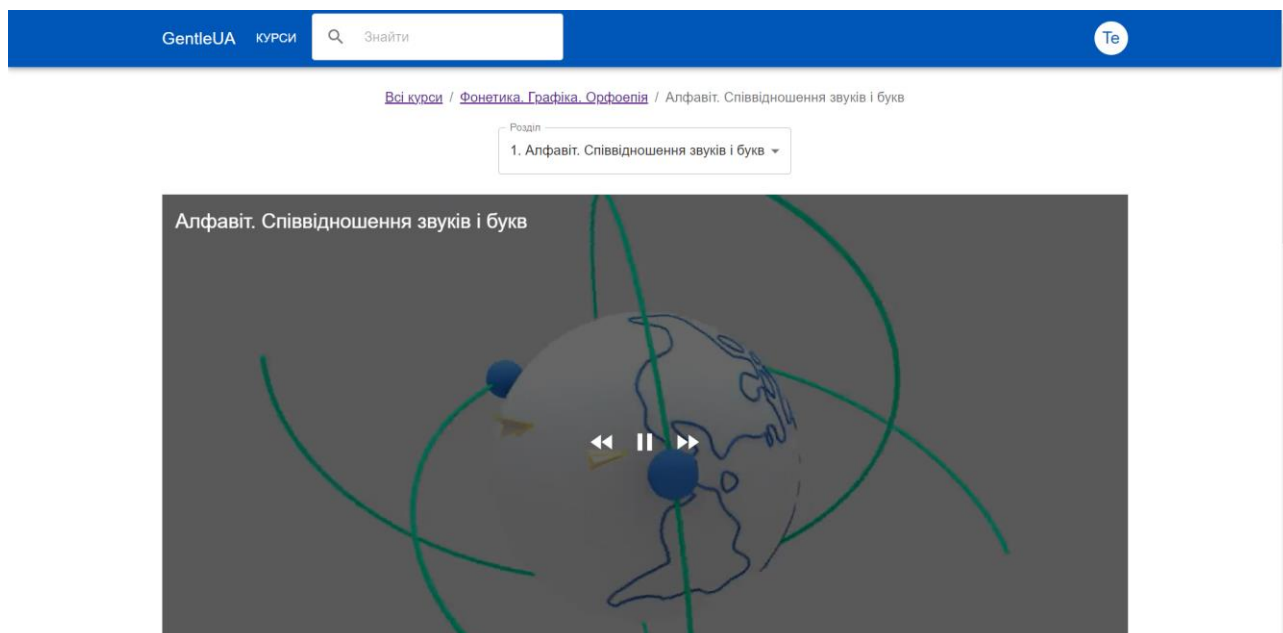


Рисунок 3.9 - Сторінка розділу “Алфавіт. Співвідношення звуків і букв”

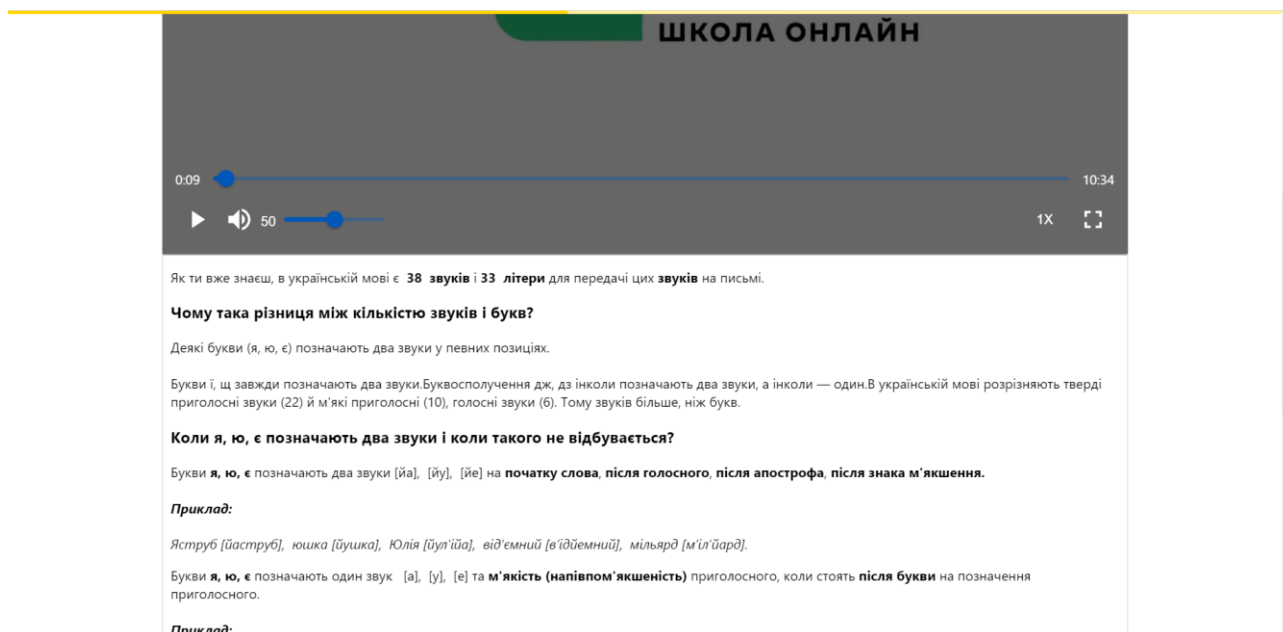


Рисунок 3.10 - Сторінка розділу “Алфавіт. Співвідношення звуків і букв”
(продовження)

Після проходження всіх розділів курсу користувачу пропонується пройти тест для перевірки отриманих знань. Після його проходження користувач дізнається свою оцінку і він може подивитися правильні та неправильні відповіді. Курс позначається завершеним і користувач має змогу залишити коментар до курсу. Тести можна проходити безліч разів, вони щоразу будуть пропонувати інші запитання. В профілі зберігається лише найкраща оцінка.

На рис. 3.11 зображено сторінку тесту для розділу, що був згаданий раніше.

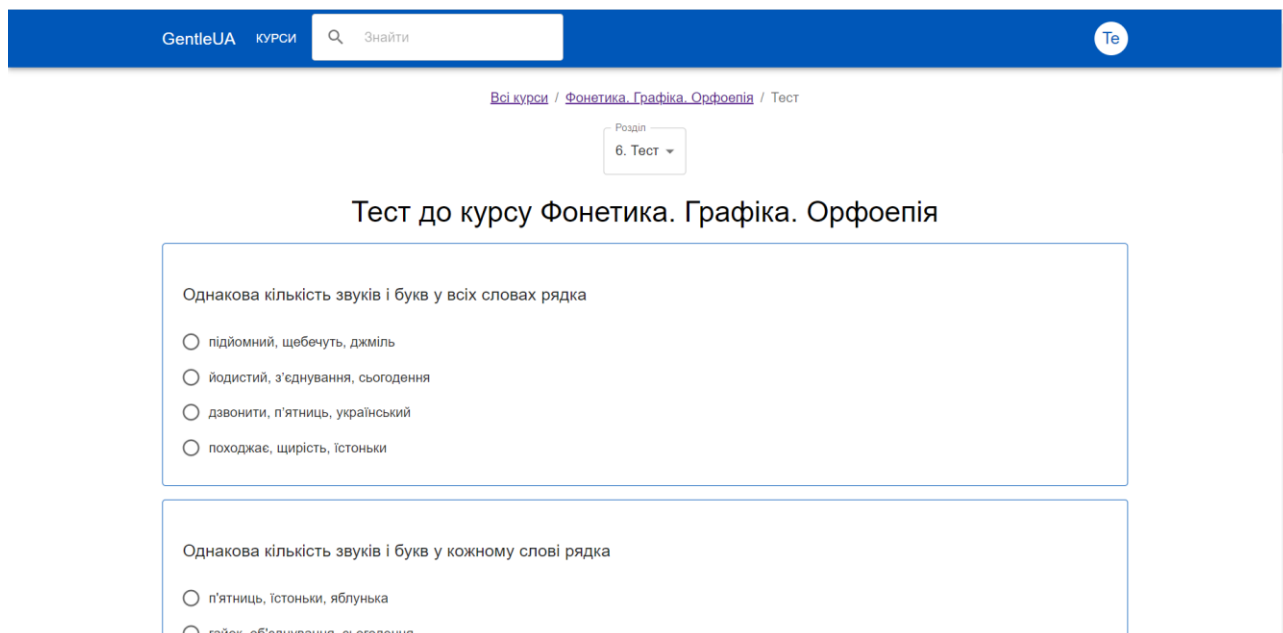


Рисунок 3.11 - Сторінка тесту

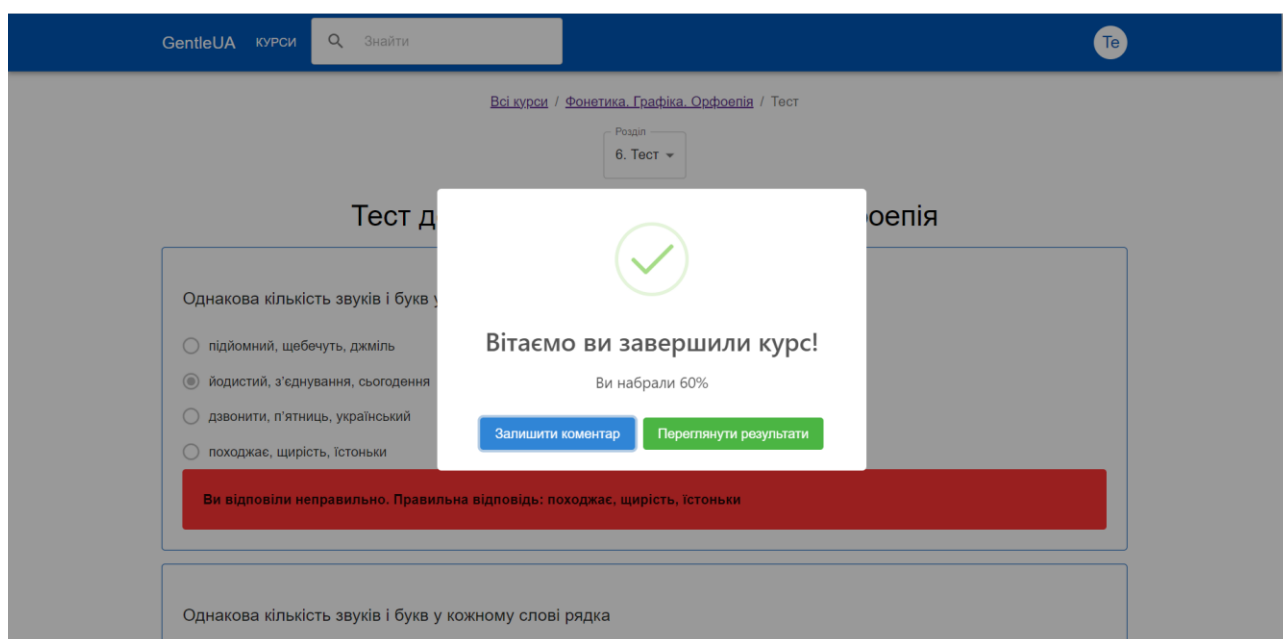


Рисунок 3.12 - Результат тесту

Також, у кожного користувача є окрема сторінка профілю, де зберігається інформація про нього та його курсу (завершені та незавершені). На завершених курсах вказано найкращу оцінку, яку отримав користувач. Також, на даній сторінці є можливість змінити пароль до свого акаунту.

На рис. 3.13 зображено сторінку профілю користувача.

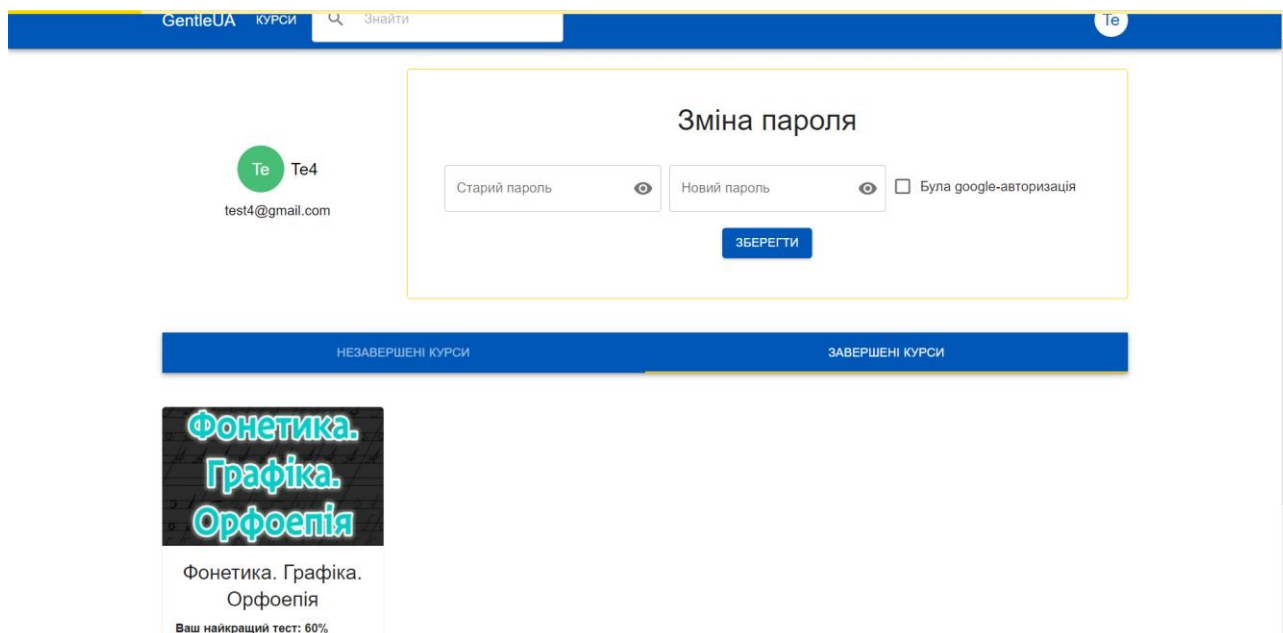


Рисунок 3.13 - Сторінка профілю користувача

В системі передбачено роль адміністратор, що відповідає за контроль вмісту на вебсайті. Користувач повинен ввести відповідні дані, щоб авторизуватися як адміністратор. На рис. 3.14 зображено адмін-панель, куди перенаправляється користувач після успішного входу.

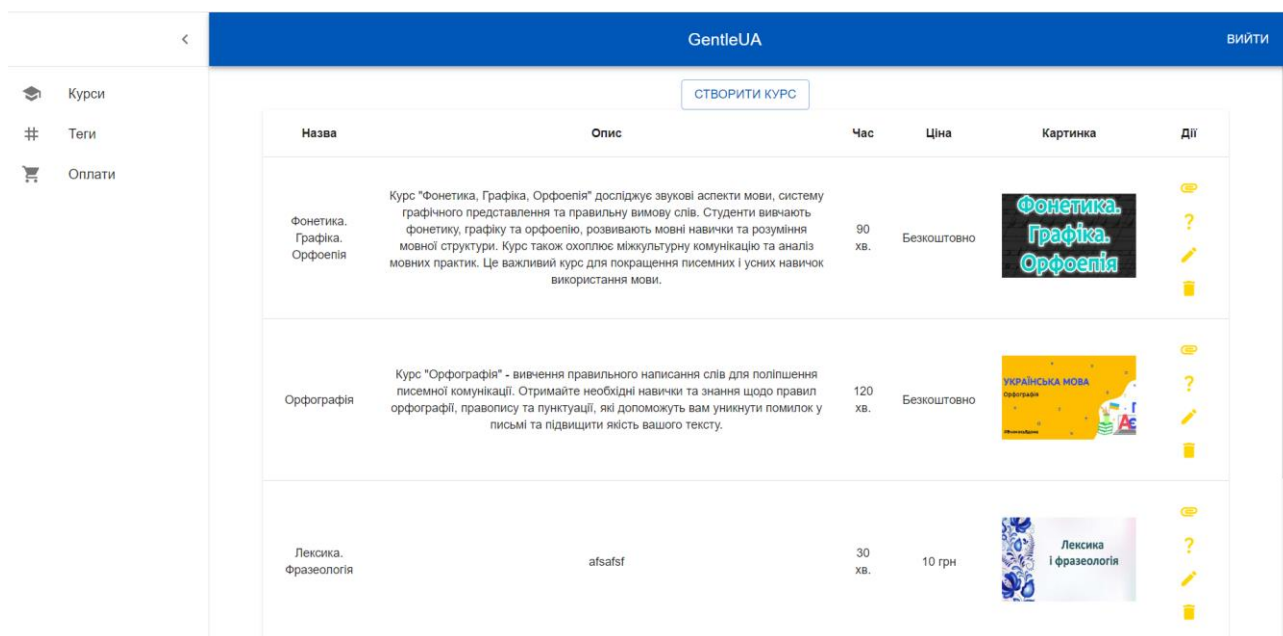


Рисунок 3.14 - Сторінка адмін-панелі

На сторінці всіх розділів курсу передбачено зміну їх порядку за допомогою відповідного користувацького інтерфейсу. На рис. 3.15 зображено даний процес.

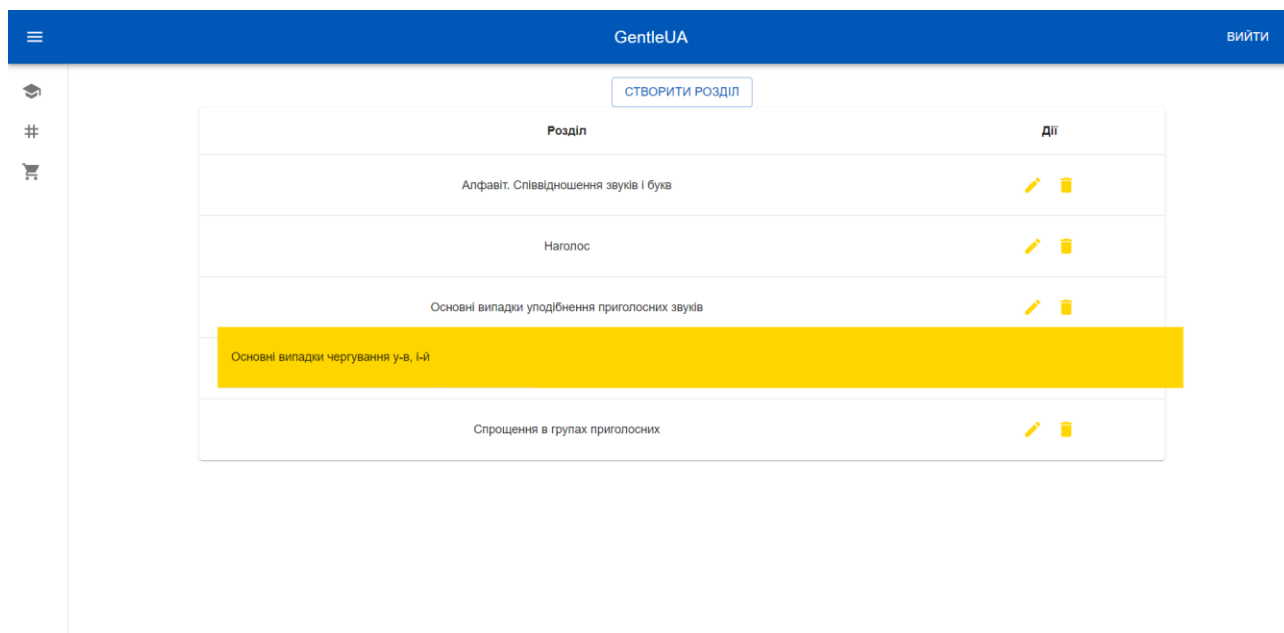


Рисунок 3.15 - Сторінка розділів певного курсу

Також, адміністратор може переглядати успішні оплати користувачів. Це продемонстровано на рис. 3.16.

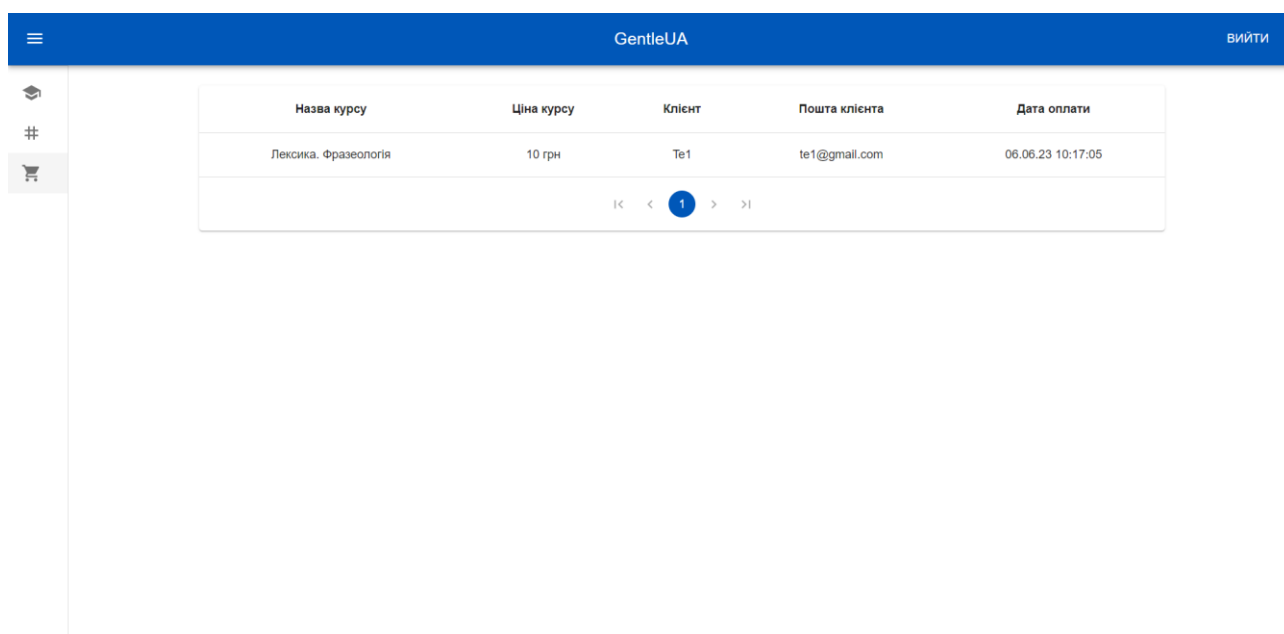


Рисунок 3.16 - Сторінка оплат користувачів

Звісно для кожної сутності системи є повний спектр керування неї, а саме: створення, редагування та видалення. Для прикладу розглянемо редагування певного розділу курсу. На цій сторінці передбачено завантаження відео на сервер з ілюстрацією процесу.

На рис. 3.17 зображено створення розділу курсу.

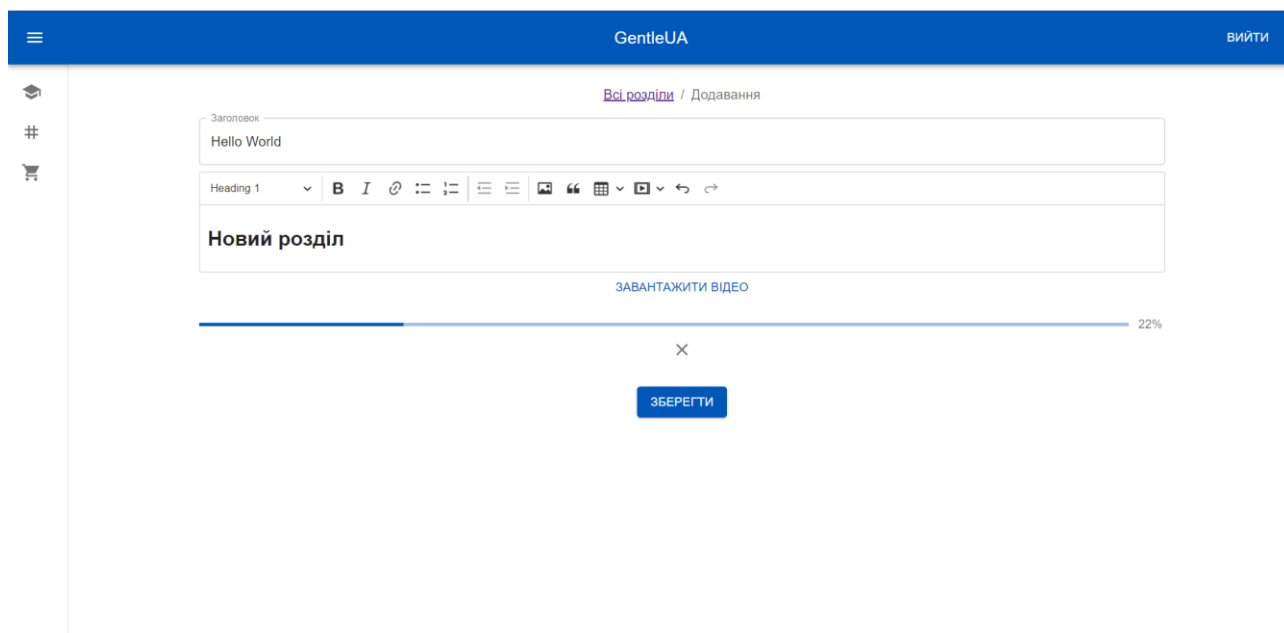


Рисунок 3.17 - Створення розділу курсу

Також, розглянемо редагування певного запитання до тесту. На цій сторінці адміністратор повинен ввести запитання і 4 варіанти відповідей, обравши з них правильний. Будь-яке запитання можна зробити неактивним і воно перестане з'являтися в тестах користувачів. Передбачено налаштування стилізації запитання для більшого спектру можливостей.

На рис. 3.18 зображено редагування певного запитання до тесту.

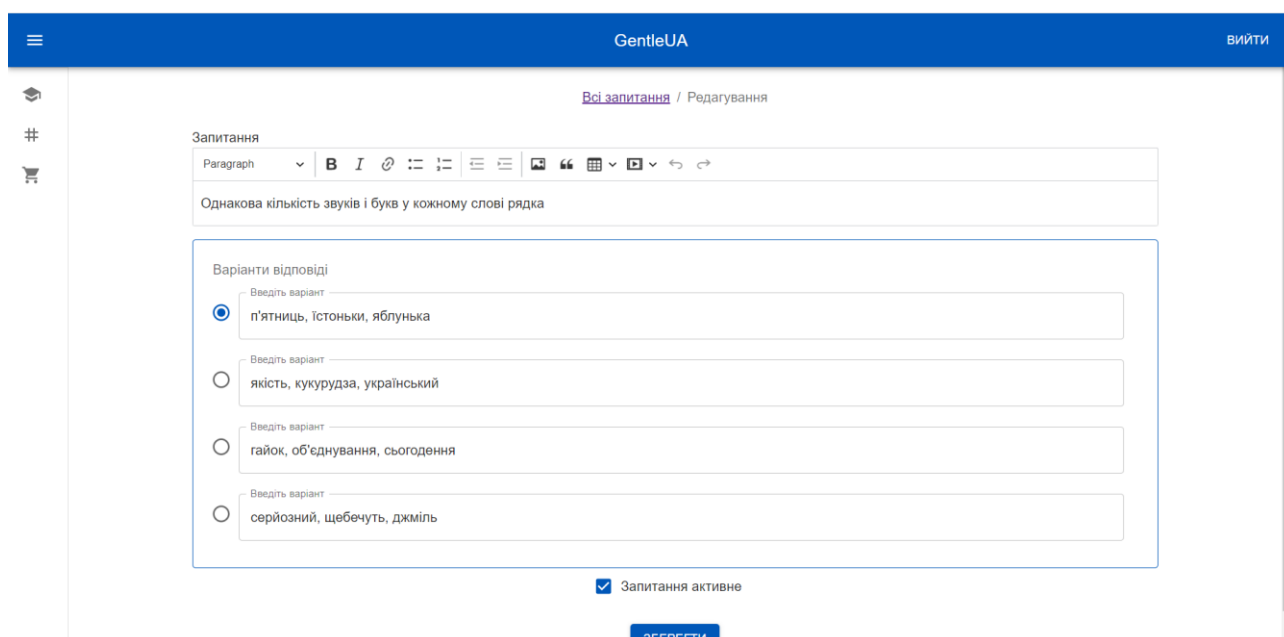


Рисунок 3.18 - Редагування певного запитання до тесту

3.3. Тестування роботи програмного продукту

Створення будь-якого програмного продукту передбачає припущення певних помилок у ході розробки, тому без тестування програмного забезпечення неможливо отримати додаток високої якості. Під час тестування перевіряються різні аспекти програми, включаючи коректність вхідних та вихідних даних, правильність роботи функцій та алгоритмів, обробка помилок і відновлення після збоїв. Метою тестування є виявлення помилок і дефектів програмного продукту, що дозволяє їх виправити перед випуском на ринок або в експлуатацію.

Для початку створимо так звані Test Cases [25]. Test Case – це тестовий артефакт, суть якого полягає у виконанні деякої кількості дій та/або умов, необхідних для перевірки певної функціональності програмної системи, що розробляється. Подамо створені тестові випадки у вигляді таблиці.

Таблиця 3.1

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
1.	2.	3.	4.	5.	6.
Перевірка авторизації з дійсними даними	1. Перейти на сторінку http://127.0.0.1:3000/login 2. Введіть email або логін 3. Введіть пароль 4. Натисніть "Увійти"	email = test@gmail.com Пароль = 12345678	Користувач повинен увійти в систему	Як і очікувалося	Pass
Перевірка авторизації з недійсними даними	1. Перейти на сторінку http://127.0.0.1:3000/login 2. Введіть email або логін 3. Введіть пароль 4. Натисніть "Увійти"	email = test@gmail.com Пароль = 12345679	Користувач не повинен входити в програму	Як і очікувалося	Pass
Перевірка реєстрації з дійсними даними	1. Перейти на сайт http://127.0.0.1:3000/register 2. Введіть логін 3. Введіть email 4. Введіть пароль 5. Підтвердіть пароль 6. Натисніть "Зареєструватися"	Логін = test email = test@gmail.com Пароль = 12345678 Підтвердження пароля = 12345678	Повинен створитися акаунт і користувач повинен увійти в систему	Як і очікувалося	Pass

Продовження табл. 3.1

1.	2.	3.	4.	5.	6.
Перевірка реєстрації з недійсними даними	1. Перейти на сайт http://127.0.0.1:3000/register 2. Введіть логін 3. Введіть email 4. Введіть пароль 5. Підтвердіть пароль 6. Натисніть "Зареєструватися"	Логін = test email = test@gmail.com Пароль = 12345678 Підтвердження пароля = 12345679	Користувач повинен отримати повідомлення про помилку, яку він припустив	Як і очікувалося	Pass
Перевірка скидання пароля	1. Перейти на сторінку http://127.0.0.1:3000/login 2. Натиснути "Забули пароль" 3. Введіть email 4. Натисніть "Скинути пароль" 5. Перейти до пошти та перейти за вказаним посиланням 6. Введіть email 7. Введіть пароль 8. Підтвердіть пароль 9. Натисніть "Змінити пароль"	Логін = test email = test@gmail.com Пароль = 12345678 Підтвердження пароля = 12345678	Пароль користувача повинен змінитися	Як і очікувалося	Pass
Перевірка додавання курсу з дійсними даними	1. Увійти в систему у ролі адміністратора 2. Перейти до адмін-панелі 3. Перейти до вкладки курси і натиснути "Створити курс" 4. Введіть дійсні дані 5. Натисніть "Зберегти"	Назва = Прийменники в українській мові Опис = Опанування теми "Прийменники в українській мові" Час проходження = 45	Повинен створитися новий курс з повідомленням про його успішне створення	Як і очікувалося	Pass
Перевірка сортування курсів	1. Увійти в систему у ролі користувача 2. Перейти на сторінку з усіма курсами 3. Змінити тип сортування за допомогою відповідного спадного списку	Критерій = За популярністю	Відображення порядку курсів повинно змінитися відповідно до вказаного критерію	Як і очікувалося	Pass

Проаналізувавши таблицю, можемо зробити висновок, що всі описані вище тестові випадки відповідають поставленим очікуванням.

Також, необхідно перевірити обмеження певного функціоналу для користувачів. Для цього скористаємося API-платформою Postman [26]. Вона дозволяє розробникам розробляти, створювати, тестувати та повторювати свої API.

На рис. 3.19 зображено спробу редагувати не свій коментар.

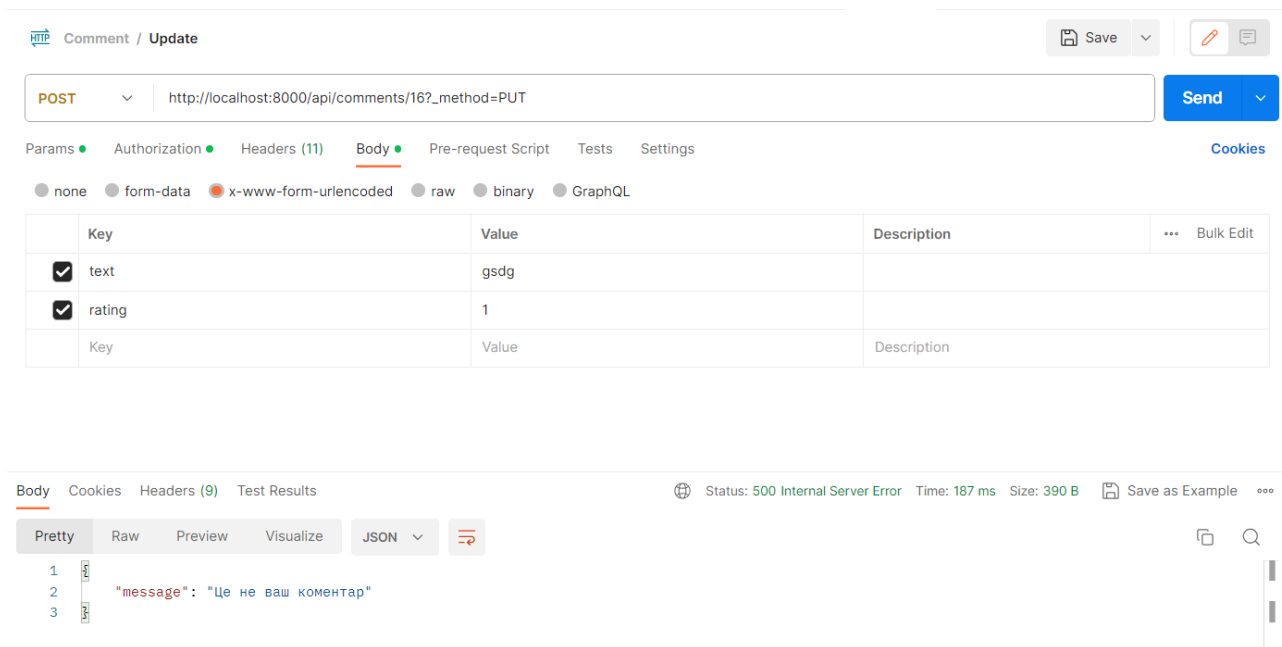


Рисунок 3.19 - Спроба редагувати свій коментар

Цю дію виконувати не вдалося, як і очікувалося. Отримано відповідне повідомлення.

Перевіримо спробу отримати розділ курсу без реєстрації на даний курс.

На рис. 3.20 зображено спробу отримати розділ курсу без реєстрації на даний курс.

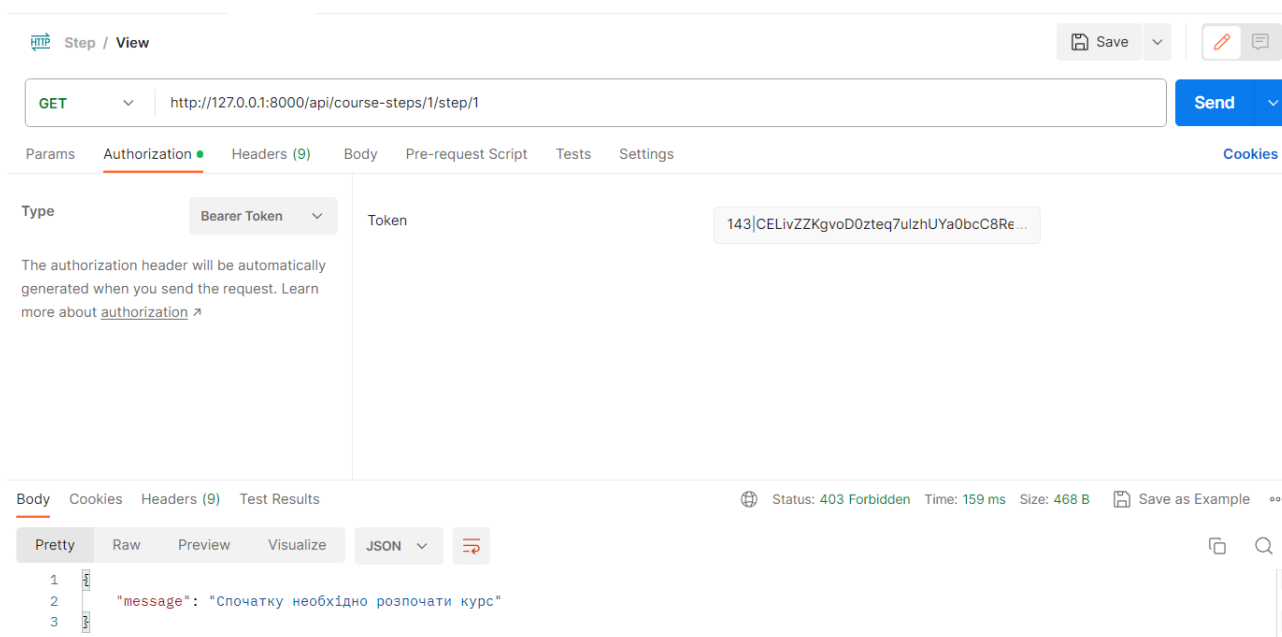


Рисунок 3.20 - Спроба отримати розділ курсу без реєстрації на даний курс

Цю дію виконувати не вдалося, як і очікувалося. Отримано відповідне повідомлення.

В ході проведення тестування було перевірено додаток та виправлено знайдені помилки. В результаті отримано застосунок відповідної якості.

Висновки до третього розділу

В даному розділі було побудовано діаграму розгортання та отримано дані щодо порядку встановлення та налаштування параметрів системи.

Також, було продемонстровано структуру інтерфейсу та порядок взаємодії з онлайн-сервісом. Було докладно описано функціональні можливості додатку в залежності від ролі користувача.

Було проведено тестування отриманого вебзастосунку за допомогою Test Cases та API-платформи Postman.

В ході проведення тестування було перевірено додаток та виправлено знайдені помилки. В результаті отримано застосунок відповідної якості.

ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано онлайн-сервіс проходження курсів української мови. Виконання поставленої задачі було розділено на 3 розділи.

У першому розділі було виконано постановку задачі на кваліфікаційну роботу. Визначено функціональні вимоги та основні етапи роботи.

Проведено аналіз аналогів до продукту, що розроблюється. І завдяки цьому отримано наступні критерії для розробки: інтуїтивний та зрозумілий дизайн; наявність відеоконтенту; наявність коментарів; українська мова інтерфейсу; достатня кількість курсів; розподіл за категоріями; наявність вправ для перевірки своїх знань.

Було проаналізовано існуючі архітектури і обрано використовувати схему SPA для забезпечення користувачу досвіду, близького до користування настільною програмою.

Також, було обґрунтовано вибір використання наступного стеку технологій для реалізації застосунку: Laravel для побудови бекенд частини, React – для фронтенд частини; в якості системи керування базами даних обрано MySQL; в якості локального вебсервера буде використовуватися OpenServer.

Були встановлені вимоги до серверного обладнання та робочої станції користувача.

В другому розділі для наочного зображення відношення між акторами та прецедентами в системі, було побудовано діаграму варіантів використання. Було сформовано вимоги до системи. Було обґрунтовано та описано використання патернів при розробці застосунку. Було спроектовано базу даних та описано поля та їх призначення.

Описано, як було використано архітектуру Single-Page Application та обґрунтовано використання підходу RESTful API. Було продемонстровано загальний алгоритм роботи та побудовані наступні діаграми для

презентативності: діаграму активностей системи, діаграму послідовності для оплати та діаграма компонентів системи.

Було продемонстровано реалізацію програмного комплексу та наведено фрагменти коду, що відповідають за авторизацію, оплату та завантаження відео на сервер з поясненням принципу їх роботи.

В третьому розділі було побудовано діаграму розгортання та отримано дані щодо порядку встановлення та налаштування параметрів системи.

Також, було продемонстровано структуру інтерфейсу та порядок взаємодії з онлайн-сервісом. Було докладно описано функціональні можливості додатку в залежності від ролі користувача.

Було проведено тестування отриманого вебзастосунку за допомогою Test Cases та API-платформи Postman.

В ході проведення тестування було перевірено додаток та виправлено знайдені помилки. В результаті отримано застосунок відповідної якості.

Розроблений програмний продуки готовий до використання. Онлайн-сервіс може бути використаної для покращення знань української мови громадян України. В майбутньому цей програмний продукт може еволюціонувати шляхом виявлення потреб у новому функціоналі, його реалізації та оновлення для всіх користувачів системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація бібліотеки React [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.legacy.reactjs.org/>.
2. Сучасний підручник з JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.javascript.info/>.
3. Документація фреймворку Laravel [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/9.x>.
4. Посібник PHP [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/manual/en/index.php>.
5. Документація MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/>.
6. Стаття про архітектуру SPA [Електронний ресурс] – Режим доступу до ресурсу: https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application?spz=learn_var.
7. Стаття про API [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.ua/news/chto-takoe-api-prostym-yazykom>.
8. Мова – ДНК нації [Електронний ресурс] – Режим доступу до ресурсу: <https://ukr-mova.in.ua/>.
9. Онлайн-платформа Udemy [Електронний ресурс] – Режим доступу до ресурсу: <https://www.udemy.com/>.
10. Вебсайт Є-Мова [Електронний ресурс] – Режим доступу до ресурсу: <https://www.emova.org.ua/>.
11. MVC Design Pattern [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/mvc-design-pattern/>.
12. Model-View-ViewModel (MVVM) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.techtarget.com/whatis/definition/Model-View-ViewModel>.
13. Документація Symfony [Електронний ресурс] – Режим доступу до ресурсу: <https://symfony.com.ua/doc/current/index.html>.

- 14.Документація Yii2 [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.yiiframework.com/doc/guide/2.0/en>.
- 15.Документація Angular [Електронний ресурс] – Режим доступу до ресурсу:
<https://angular.io/docs>.
- 16.Документація Vue.js [Електронний ресурс] – Режим доступу до ресурсу:
<https://vuejs.org/guide/introduction.html>.
- 17.Use Case Diagram [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>.
- 18.Class Diagram [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>.
- 19.Laravel API — Repository Pattern [Електронний ресурс] – Режим доступу до ресурсу:
<https://blog.devgenius.io/laravel-api-repository-pattern-make-your-code-more-structured-the-simple-guide-5b770da766d7>.
- 20.What is REST? [Електронний ресурс] – Режим доступу до ресурсу:
<https://restfulapi.net/>.
- 21.Activity Diagram [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>.
- 22.Component Diagram [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/>.
- 23.Sequence Diagram [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>.
- 24.Deployment Diagram [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/>.

25. Test case [Електронний ресурс] – Режим доступу до ресурсу:
<https://qalight.ua/baza-znaniy/test-case-2/>.
26. API-платформа Postman [Електронний ресурс] – Режим доступу до
ресурсу: <https://www.postman.com/>.

ДОДАТКИ

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	
						61

Клас, що відповідає за завантаження відео на сервер

```

<?php

namespace App\Repositories;

use App\Interfaces\BigFileInterface;
use App\Traits\FileUpload;
use App\Traits\ResponseApi;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\UploadedFile;
use Illuminate\Support\Facades\App;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Validator;
use Pion\Laravel\ChunkUpload\Exceptions\UploadFailedException;
use Pion\Laravel\ChunkUpload\Exceptions\UploadMissingFileException;
use Pion\Laravel\ChunkUpload\Handler\HandlerFactory;
use Pion\Laravel\ChunkUpload\Receiver\FileReceiver;

class BigFileRepository implements BigFileInterface
{
    use ResponseApi, FileUpload;

    public string $save_path = 'public/uploadedVideos';

    /**
     * @param Request $request
     * @return JsonResponse
     */
    public function destroy(Request $request): JsonResponse
    {
        if (Storage::delete('/public/uploadedVideos/' . $request->file))
            return $this->success('Файл успішно видалено');
        return $this->error('Сталася помилка видалення');
    }

    /**
     * @param Request $request
     * @return JsonResponse
     * @throws UploadMissingFileException
     * @throws UploadFailedException
     */
    public function uploadFile(Request $request): JsonResponse
    {
        $receiver = new FileReceiver('file', $request,
            HandlerFactory::classFromRequest($request));

        if ($receiver->isUploaded() === false) {
            throw new UploadMissingFileException();
        }

        $save = $receiver->receive();

        if ($save->isFinished()) {
            return $this->saveFile($save->getFile());
        }

        $handler = $save->handler();
        return $this->success('Chunk успішно передано', ['done' => $handler->getPercentageDone(),
            'status' => true]);
    }

    protected function saveFile(UploadedFile $file): JsonResponse
    {
        $fileName = $this->createFilename($file);
    }
}

```

```

        $mime = str_replace('/', '-', $file->getMimeType());
        $path = Storage::putFileAs($this->save_path, $file, $fileName);

        $path = preg_replace('/public/', '', $path);
        unlink($file->getPathname());

        return $this->success('Файл успішно збережено', [
            'complete' => true,
            'path' => @App::make('url')->to('/') . '/storage' . $path,
            'name' => $fileName,
            'mime_type' => $mime
        ]);
    }
}

```

Клас, що відповідає за тестування в системі

```

<?php

namespace App\Repositories;

use App\Http\Requests\QuestionRequest;
use App\Interfaces\QuestionInterface;
use App\Models\Course;
use App\Models\Order;
use App\Models\Question;
use App\Models\QuestionChoice;
use App\Models\User;
use App\Models\UserCourse;
use App\Traits\ResponseApi;
use Illuminate\Database\Eloquent\ModelNotFoundException;
use Illuminate\Http\Request;
use Illuminate\Http\JsonResponse;
use Symfony\Component\HttpFoundation\Response;

class QuestionRepository implements QuestionInterface
{
    use ResponseApi;

    public function store(QuestionRequest $request, int $course_id): JsonResponse
    {
        try {
            $question = Question::create([
                'question' => $request->question,
                'is_active' => $request->is_active,
                'course_id' => $course_id
            ]);

            foreach ($request->choices as $id => $choice) {
                $question->choices()->create([
                    'choice' => $choice,
                    'is_right' => $id == $request->is_right ? 1 : 0,
                    'question_id' => $question->id
                ]);
            }

            return $this->success('Запитання успішно створено', ['question' => $question]);
        } catch (\Throwable $th) {
            return $this->error($th->getMessage());
        }
    }
}

```

```

    }

    public function update(QuestionRequest $request, int $id): JsonResponse
    {
        try {
            $question = Question::findOrFail($id);
            $question->update([
                'question' => $request->question,
                'is_active' => $request->is_active,
                'course_id' => $question->course_id
            ]);

            foreach ($question->choices as $id => $choice) {
                $choice->choice = $request->choices[$id];
                $choice->is_right = $id == $request->is_right ? 1 : 0;
                $choice->save();
            }

            return $this->success('Запитання успішно відредаговано', ['question' => $question]);
        } catch (ModelNotFoundException $e) {
            return $this->error($e->getMessage(), Response::HTTP_NOT_FOUND);
        } catch (\Throwable $th) {
            return $this->error($th->getMessage());
        }
    }

    public function index(Request $request): JsonResponse
    {
        try {
            if (!Order::where(['course_id' => $request->course_id, 'user_id' => auth()->id(),
                'status' => 'success'])->first() &&
                !User::find(auth()->id()->isAdmin() && Course::findOrFail($request->course_id)-
                >price) {
                return $this->error('Спочатку необхідно купити курс', Response::HTTP_FORBIDDEN);
            }

            if (!UserCourse::where(['course_id' => $request->course_id, 'user_id' => auth()-
                >id()])->first() &&
                !User::find(auth()->id()->isAdmin()) {
                return $this->error('Спочатку необхідно розпочати курс',
                Response::HTTP_FORBIDDEN);
            }

            $defaultPerPage = 10;
            if ($request->random) {
                $questions = Question::with(['choices' => function ($q) {

```

```

        $q->inRandomOrder();
    }])
    ->where(['is_active' => 1, 'course_id' => $request->course_id])
    ->inRandomOrder()->limit(10)->get();
    } else {
        $questions = Question::where(['course_id' => $request->course_id])-
>paginate($request->per_page ?? $defaultPerPage);
    }

    return $this->success('Запитання успішно отримані', ['questions' => $questions]);
} catch (ModelNotFoundException $e) {
    return $this->error($e->getMessage(), Response::HTTP_NOT_FOUND);
} catch (\Throwable $th) {
    return $this->error($th->getMessage());
}
}

public function show(int $id): JsonResponse
{
    try {
        $question = Question::with('choices')->findOrFail($id);

        $question->choices->makeVisible('is_right');

        return $this->success('Запитання успішно отримано', ['question' => $question]);
    } catch (ModelNotFoundException $e) {
        return $this->error($e->getMessage(), Response::HTTP_NOT_FOUND);
    } catch (\Throwable $th) {
        return $this->error($th->getMessage());
    }
}

public function destroy(int $id): JsonResponse
{
    try {
        $question = Question::findOrFail($id);
        $question->delete();

        return $this->success('Запитання успішно видалено');
    } catch (ModelNotFoundException $e) {
        return $this->error($e->getMessage(), Response::HTTP_NOT_FOUND);
    } catch (\Throwable $th) {
        return $this->error($th->getMessage());
    }
}
}

```

```

public function checkAnswers(Request $request): JsonResponse
{
    try {
        if (!$userCourse = UserCourse::where(['course_id' => $request->all()[0]['course_id'],
'user_id' => auth()->id()])->first()) {
            return $this->error('Спочатку необхідно розпочати курс',
Response::HTTP_FORBIDDEN);
        }
        $answers = [];
        $right = 0;
        foreach ($request->all() as $id => $item) {
            $choice = QuestionChoice::findOrFail($item['choice']['id']);
            $rightAnswer = QuestionChoice::where(['is_right' => 1, 'question_id' =>
$item['id']])->first();
            if ($choice->is_right) {
                $answers['res'][$id] = ['res' => true];
                $right++;
            } else {
                $answers['res'][$id] = ['res' => false, 'right' => $rightAnswer->choice];
            }
        }
        $total = round($right / count($request->all()) * 100);
        $answers['total'] = $total;
        if ($userCourse->grade < $total || !$userCourse->grade) {
            $userCourse->grade = $total;
        }
        $userCourse->is_finished = 1;
        $userCourse->save();
        return $this->success('Відповіді успішно опрацьовано', ['answers' => $answers]);
    } catch (\Throwable $th) {
        return $this->error($th->getMessage());
    }
}
}

```

Клас, що відповідає за авторизацію

```
<?php

namespace App\Repositories;

use App\Http\Requests\ForgotPasswordRequest;
use App\Http\Requests\RegisterRequest;
use App\Http\Requests\ResetPasswordRequest;
use App\Interfaces\AuthInterface;
use App\Mail\ForgotMail;
use App\Models>PasswordReset;
use App\Models\User;
use App\Traits\ResponseApi;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Mail;
use Symfony\Component\HttpFoundation\Response;

class AuthRepository implements AuthInterface
{
    use ResponseApi;

    /**
     * Create User
     * @param RegisterRequest $request
     * @return JsonResponse
     */
    public function register(RegisterRequest $request): JsonResponse
    {
        try {
            $user = User::create([
                'name' => $request->name,
                'email' => $request->email,
                'password' => Hash::make($request->password),
                'color' => $this->generateRandomColor()
            ]);
            $token = $user->createToken($user->name . '_Token', [''])->plainTextToken;
            return $this->success('Акаунт успішно створено', ['token' => $token, 'user' =>
            $user]);
        } catch (\Throwable $th) {
            return $this->error($th->getMessage());
        }
    }
}
```

```

    }

    /**
     * Login User
     * @param Request $request
     * @return JsonResponse
     */
    public function login(Request $request): JsonResponse
    {
        $login = $request->login;
        $fieldType = filter_var($login, FILTER_VALIDATE_EMAIL) ? 'email' : 'name';
        request()->merge([$fieldType => $login]);
        try {
            if (!Auth::attempt($request->only([$fieldType, 'password']))) {
                return $this->error(
                    'Не знайдено користувача з вказаними даними.',
                    Response::HTTP_UNAUTHORIZED);
            }

            $user = auth()->user();

            if ($user->isAdmin()) {
                $token = $user->createToken($user->name . '_AdminToken', ['server:admin'])-
>plainTextToken;
                $user->role = 1;
            } else {
                $token = $user->createToken($user->name . '_Token', [''])->plainTextToken;
                $user->role = 0;
            }

            return $this->success('Ви успішно увійшли в систему', ['token' => $token, 'user' =>
$user]);

        } catch (\Throwable $th) {
            return $this->error($th->getMessage());
        }
    }

    /**
     * Logout User
     * @return JsonResponse
     */
    public function logout(): JsonResponse
    {
        auth()->user()->tokens()->delete();
        return $this->success('Ви успішно вийшли з системи');
    }

```

```

}

/**
 * Generate token for reset
 * @param ForgotPasswordRequest $request
 * @return JsonResponse
 */
public function forgotPassword(ForgotPasswordRequest $request): JsonResponse
{
    $email = $request->email;
    $user = User::where('email', $email);

    if ($user->doesntExist()) {
        return $this->error('Не існує користувача з вказаним email',
Response::HTTP_UNAUTHORIZED);
    }

    $token = rand(10, 100000);
    try {
        if (!$userPassReset = PasswordReset::where('email', $email)->first()) {
            PasswordReset::create([
                'email' => $email,
                'token' => $token
            ]);
        } else {
            $userPassReset->update([
                'email' => $email,
                'token' => $token
            ]);
        }
        Mail::to($email)->send(new ForgotMail($token, $user->first()->name));

        return $this->success("Лист надіслано на пошту " . $email);
    } catch (\Throwable $th) {
        return $this->error($th->getMessage());
    }
}

/**
 * Reset User's password
 * @param ResetPasswordRequest $request
 * @return JsonResponse
 */
public function resetPassword(ResetPasswordRequest $request): JsonResponse
{
    $user = User::where('email', $request->email)->first();

```

```

        if (!$user) {
            return $this->error('Не існує користувача з вказаним email',
Response::HTTP_UNAUTHORIZED);
        }

        $resetRequest = PasswordReset::where('email', $user->email)->first();

        if (!$resetRequest || $resetRequest->token != $request->token) {
            return $this->error('Не дійсний токен скидання', Response::HTTP_BAD_REQUEST);
        }

        try {
            $user->fill([
                'password' => Hash::make($request->password)
            ]);
            $user->save();
            $user->tokens()->delete();
            $resetRequest->delete();
            return $this->success("Пароль успішно змінено");
        } catch (\Throwable $th) {
            return $this->error($th->getMessage());
        }
    }

    public function changePassword(Request $request)
    {
        try {
            $user = User::findOrFail(auth()->id());

            if ($request->is_google && $user->password) {
                return $this->error("У вас вже встановлений старий пароль");
            }

            if (!Hash::check($request->old_password, $user->password)) {
                return $this->error("Неправильний старий пароль");
            }

            $user->fill([
                'password' => Hash::make($request->new_password)
            ]);
            $user->save();

            return $this->success("Пароль успішно змінено");
        } catch (\Throwable $th) {
            return $this->error($th->getMessage());
        }
    }

```

```

    }
}

/**
 * Return random color
 *
 * @return string
 */
private function generateRandomColor(): string
{
    $red = dechex(rand(0, 200));
    $green = dechex(rand(0, 200));
    $blue = dechex(rand(0, 200));

    $lightness = (hexdec($red) + hexdec($green) + hexdec($blue)) / 3;

    if ($lightness > 180) {
        return $this->generateRandomColor();
    }
    return '#' . $red . $green . $blue;
}
}

```

ОСНОВНИЙ КОМПОНЕНТ КЛІЄНТСЬКОЇ ЧАСТИНИ

```

import './App.css';
import {createBrowserRouter, RouterProvider} from "react-router-dom";
import GoogleLogin from "../pages/Authorization/GoogleLogin";
import {LoginForm} from "../pages/Authorization/LoginForm/LoginForm";
import {RegisterForm} from "../pages/Authorization/RegisterForm/RegisterForm";
import {ResetForm} from "../pages/Authorization/ResetForm/ResetForm";
import Layout from "../components/layouts/Layout";
import React from "react";
import ProtectedRoute from "../components/routes/ProtectedRoute";
import Courses from "../pages/Course/Courses/Courses";
import AdminProtectedRoute from "../components/routes/AdminProtectedRoute";
import CourseForm from "../pages/Course/CourseForm/CourseForm";
import CoursesIndex from "../pages/Course/CoursesIndex/CoursesIndex";
import Forbidden from "../pages/Errors/Forbidden/Forbidden";
import NotFound from "../pages/Errors/NotFound/NotFound";
import CourseView from "../pages/Course/CourseView/CourseView";
import CourseStepForm from "../pages/CourseStep/CourseStepForm/CourseStepForm";
import CourseStepView from "../pages/CourseStep/CourseStepView/CourseStepView";
import CourseStepsIndex from "../pages/CourseStep/CourseStepIndex/CourseStepsIndex";
import TagForm from "../pages/Tag/TagForm/TagForm";
import TagIndex from "../pages/Tag/TagIndex/TagIndex";
import AdminLayout from "../components/layouts/AdminLayout";
import Profile from "../pages/Authorization/Profile/Profile";
import QuizForm from "../pages/Quiz/QuizForm/QuizForm";
import QuizIndex from "../pages/Quiz/QuizIndex/QuizIndex";
import QuizView from "../pages/Quiz/QuizView/QuizView";
import {HelmetProvider} from "react-helmet-async";
import OrderIndex from "../pages/Order/OrderIndex";
import Main from "../pages/Main/Main";

function App() {
  const router = createBrowserRouter([
    {
      path: "/",
      element: <Layout/>,
      children: [
        {
          path: "/",
          element: <Main/>
        },
        {
          path: "/login",
          element: <LoginForm/>
        },
        {
          path: "/register",
          element: <RegisterForm/>
        },
        {
          path: "/reset-password/:token",
          element: <ResetForm/>
        },
        {
          path: "/auth/login",
          element: <GoogleLogin/>
        },
        {
          path: "/error/403",
          element: <Forbidden/>
        },
        {
          path: "/error/404",
          element: <NotFound/>
        }
      ]
    }
  ]),

```

```

        {
            path: "*",
            element: <NotFound/>
        }
    ],
},
{
    path: "/",
    element: <ProtectedRoute><Layout/></ProtectedRoute>,
    children: [
        {
            path: "/courses",
            element: <Courses/>,
        },
        {
            path: "/course/:id",
            element: <CourseView/>
        },
        {
            path: "/course/:course_id/step/:number_step",
            element: <CourseStepView/>
        },
        {
            path: "/profile",
            element: <Profile/>
        },
        {
            path: "/course/:course_id/quiz",
            element: <QuizView/>
        }
    ]
},
{
    path: "/",
    element: <AdminProtectedRoute><AdminLayout/></AdminProtectedRoute>,
    children: [
        {
            path: "/course/add",
            element: <CourseForm/>
        },
        {
            path: "/tag/add",
            element: <TagForm/>
        },
        {
            path: "/tag/update/:id",
            element: <TagForm/>
        },
        {
            path: "/tags",
            element: <TagIndex/>
        },
        {
            path: "/course/:course_id/step/update/:number_step",
            element: <CourseStepForm/>
        },
        {
            path: "/course/:course_id/step/add/",
            element: <CourseStepForm/>
        },
        {
            path: "/course/:course_id/steps",
            element: <CourseStepsIndex/>
        },
        {
            path: "/courses/index",
            element: <CoursesIndex/>
        },
    ]
}

```

```

        path: "/course/update/:id",
        element: <CourseForm/>
      },
      {
        path: "/course/:course_id/quiz/update/:id",
        element: <QuizForm/>
      },
      {
        path: "/course/:course_id/quiz/add",
        element: <QuizForm/>
      },
      {
        path: "/course/:course_id/quizzes",
        element: <QuizIndex/>
      },
      {
        path: "/orders",
        element: <OrderIndex/>
      },
    ],
  });

  return (
    <div className="App">
      <HelmetProvider>
        <RouterProvider router={router}/>
      </HelmetProvider>
    </div>
  );
}

export default App;

```