КВАЛІФІКАЦІЙНА РОБОТА

Група ІПЗс-20
Кердяк В.Г.

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних і прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Кердяк Володимир Григорович

УДК 004.4

Розробка веб-сайту для управління освітніми матеріалами та курсами навчання

Спеціальність 121 - «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття освітнього ступеню бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Кердяк В.Г.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

к.т.н., проф. каф. ІТ

_____ Пашкевич О.П.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО «УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА»

Факультет суспільних і прикладних наук

Кафедра інформаційних технології

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«___»_____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кердяк Володимир Григорович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Розробка веб-сайту для управління освітніми матеріалами та курсами навчання

Керівник роботи: Пашкевич Олег Петрович, к.т.н., проф. каф. ІТ

Затверджена наказом вищого навчального закладу від «12» березня 2024 року № 19/1

2. Термін подання студентом роботи: 05.06.2024

3. Вихідні дані роботи: Серверна аплікація, клієнтська частина

4. Зміст кваліфікаційної роботи (перелік питань які потрібно розробити):

1. Аналіз існуючих рішень для управління освітніми матеріалами

2. Вибір технологій для розробки

3. Розробка серверної аплікації та клієнстької частини

5. Дата видачі завдання: 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналаіз сервісів для керування освітніми матеріалами	28.03.2024	Виконано
2.	Дослідження технологій для створення системи	04.04.2024	Виконано
3.	Розробка системи для керування освітніми матеріалами	11.04.2024	Виконано
4.	Розміщення системи в мережу “Інтернет”	18.04.2024	Виконано
5.	Формування висновків	25.04.2024	Виконано
6.	Оформлення пояснювальної записки	05.05.2024	Виконано
7.	Оформлення графічного матеріалу та підготовка до захисту роботи	10.05.2024	Виконано

Студент _____ Кердяк В.Г.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Пашкевич О.П.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Сторінка	Опис граф. матеріалу	Сторінка	Опис граф. матеріалу
14	Головна сторінка веб-сайту prometheus.org.ua	49	Сторінка курсів
15	Головна сторінка веб-сайту ed-era.com	50	Сторінка деталей курсу
17	Головна сторінка веб-сайту vumonline.ua	51	Головна панель адміністратора

18	Головна сторінка веб-сайту WiseCow.com.ua	52	Розділ користувачі
27	Схема компілятора TypeScript, який перетворює код TypeScript в JavaScript, а також деякі файли визначення типів або вихідні карти	52	Процес створення курсу
28	Паттерн Controller-Service-Repository	53	Розділ з інформацією про курси
44	Інтерфейс Postman з колекцією CapyLearmo	53	Розділ команда
47	Головна сторінка веб-додатку CapyLearmo	55	веб-інтерфейс для підключення репозиторію
48	Сторінка профілю користувача		

АНОТАЦІЯ

У кваліфікаційній роботі розглянуто розробку веб-сайту для керування освітніми матеріалами та навчальними курсами з використанням сучасних веб-технологій. Розроблений сайт надає зручний доступ до освітніх ресурсів для користувачів, дозволяючи переглядати, редагувати та видаляти навчальні матеріали. Крім того, сайт сприяє підвищенню ефективності навчальних процесів і покращує користувацький досвід на платформі.

У роботі детально розглянуто процес розробки веб-сайту, включаючи аналіз існуючих платформ, проектування архітектури та дизайну сайту, реалізацію функціональності, тестування та впровадження. Було використано сучасні технології, такі як TypeScript, Node.js для серверної частини та Next.js для клієнтської частини.

Сайт надає користувачам можливість зручно керувати своїми освітніми матеріалами, забезпечуючи гнучкість та контроль над навчальними курсами. Також він сприяє ефективній взаємодії між користувачами, спрощуючи процес організації освітнього процесу.

Результати роботи показали, що розроблений сайт успішно впроваджує функціональність для управління освітніми ресурсами, сприяючи підвищенню ефективності навчання та доступності освітніх послуг.

КЛЮЧОВІ СЛОВА: TYPESCRIPT, TAILWIND, HTML, NODEJS, NEXTJS

SUMMARY

The qualification work considers the development of a website for managing educational materials and training courses using modern web technologies. The developed website provides convenient access to educational resources for users, allowing them to view, edit, and delete educational materials. In addition, the site helps to increase the efficiency of educational processes and improves the user experience on the platform.

The paper describes in detail the process of website development, including analysis of existing platforms, design of the website architecture and design, implementation of functionality and testing. Modern technologies were used, such as TypeScript, Node.js for the server side and Next.js for the client side.

The site provides users with the ability to manage their educational materials, providing flexibility and control over training courses. It also promotes effective interaction between users, simplifying the process of organizing the educational process.

The results of the work have shown that the developed website successfully implements the functionality for managing educational resources, contributing to the efficiency of learning and the availability of educational services.

KEYWORDS: TYPESCRIPT, TAILWIND, HTML, NODEJS, NEXTJS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ....9
РОЗДІЛ 1. АНАЛІЗ ПОТРЕБ У СИСТЕМАХ КЕРУВАННЯ ОСВІТНІМИ МАТЕРІАЛАМИ ТА НАВЧАЛЬНИМИ КУРСАМИ, ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ....12
1.1 Огляд існуючих систем керування освітніми матеріалами....12
1.2 Порівняльний аналіз розглянутих програм-аналогів....19
1.3 Постановка задачі....21
Висновки до розділу 1....22
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ОСВІТНІМИ МАТЕРІАЛАМИ....24
2.1 Архітектура та технологічна реалізація системи керування освітніми матеріалами....24
2.2 TypeScript та його використання у розробці серверної частини....25
2.3 Архітектура серверної частини....27
2.4 Node.js та його використання у розробці серверної частини....28
2.5 Обґрунтування використання MongoDB у системі керування освітніми матеріалами та навчальними курсами....29
2.7 Tailwind CSS....31
2.8 Redux....32
2.9 Проектування архітектури серверної частини....32
Висновки до розділу 2....33
РОЗДІЛ 3. РОЗРОБКА СЕРВЕРНОЇ ТА КЛІЄНСТЬКОЇ ЧАСТИНИ, ЇХ РОЗМІЩЕННЯ В ІНТЕРНЕТ....35
3.1 Розробка серверної частини....35

3.1.1 Розробка моделей 35
3.1.2 Використання Next.js 36
3.1.3 Розробка сервісів 37
3.1.4 Розробка контролерів 38
3.1.5 Розробка utils, middleware 39
3.1.6 Тестування серверної частини 41
3.1.7 Розміщення серверної частини в мережу Інтернет 43
3.2 Розробка клієнстької частини 44
3.2.1 Сторінки проекту 45
3.2.2 Розміщення для загального доступу 52
Висновки до розділу 3 54
ВИСНОВКИ 56
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 57
ДОДАТКИ 59
Додаток А 59
Додаток Б 64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheets
JS	–	JavaScript
МВОК	–	Масові відкриті онлайн-курси
API	–	Application Programming Interface

ВСТУП

Актуальність теми. В умовах швидкого розвитку інформаційних технологій освіта переживає період цифрової трансформації, що відкриває нові можливості для навчання та саморозвитку. Електронні освітні ресурси стають важливими для індивідуального та корпоративного навчання. Основним викликом є створення доступних та інтуїтивно зрозумілих інструментів для управління навчальними матеріалами та курсами.

Мета роботи. Розробка веб-сайту для ефективного управління освітніми матеріалами та навчальними курсами.

Об’єкт роботи. Онлайн-платформа для створення та управління освітніми матеріалами.

Предмет роботи. Програмна архітектура та інтерфейс користувача веб-сайту для управління освітніми курсами.

Завдання роботи. Відповідно до вибраної теми в роботі були покладені наступні задачі:

- дослідження потреб користувачів та аналіз існуючих рішень у галузі управління освітніми ресурсами;
- вибір мови програмування та технологій розробки;
- розробка клієнтської частини з використанням Next.js та серверної частини на платформі Node.js;
- тестування та оптимізація функціоналу веб-сайту.

Методи роботи. Для вирішення поставлених завдань були використані такі технології: програмування на TypeScript з використанням Node.js для розробки бекенду, Next.js для фронтенду, база даних MongoDB, а також методи роботи з базами даних і веб-API.

Результати роботи. В результаті виконання кваліфікаційної роботи був створений повнофункціональний веб-сайт, який дозволяє користувачам ефективно управляти навчальними матеріалами та курсами, здійснювати їх організацію та поширення серед зацікавлених осіб. Сайт містить інтуїтивно зрозумілий інтерфейс та ряд інноваційних функцій, що полегшують процес навчання та викладання.

Структура роботи. Розділи – 3. Загальний обсяг основної частини – 46 сторінок. Список використаних джерел – 20.

РОЗДІЛ 1. АНАЛІЗ ПОТРЕБ У СИСТЕМАХ КЕРУВАННЯ ОСВІТНІМИ МАТЕРІАЛАМИ ТА НАВЧАЛЬНИМИ КУРСАМИ, ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд існуючих систем керування освітніми матеріалами

В мережі Інтернет можна знайти безліч сайтів, присвячених наданню інформації про освітні платформи, навчальні курси та викладачів. Проте, серед цих сайтів виявляється значна кількість, які не пропонують можливість ефективного управління навчальними матеріалами та курсами, і це є серйозним недоліком, який впливає на зручність та ефективність для користувачів системи.

Відсутність інтегрованої системи управління навчальними матеріалами ускладнює процес організації та планування навчального процесу для викладачів та студентів. Користувачам доводиться вручну відстежувати прогрес, завантажувати матеріали окремо або використовувати кілька різних платформ для доступу до курсів та навчальних ресурсів. Це може вимагати додаткових зусиль і часу з боку користувачів, особливо у випадках, коли вони мають обмежений час або потребують швидкого доступу до матеріалів.

Брак інтегрованої системи управління також може призводити до плутанини та зниження ефективності навчального процесу. Користувачі можуть втратити мотивацію або зазнати труднощів у навчанні через незручний доступ до матеріалів та відсутність централізованого місця для зберігання та організації курсів. Це може створювати негативне враження про платформу та призвести до втрати потенційних користувачів.

Ефективна система управління навчальними матеріалами та курсами є важливою функцією, яка сприяє зручності та ефективності для користувачів. Це дає можливість користувачам самостійно організовувати та відстежувати свій

навчальний процес, отримувати доступ до необхідних ресурсів в будь-який час та з будь-якого місця без необхідності використовувати кілька різних платформ для навчання нових навичків.

Сайти, які було проаналізовано:

– prometheus.org.ua;
– vumonline.ua;
– ed-era.com;
– wisecow.com.ua.

Сайт prometheus.org.ua [17] є однією з провідних українських платформ для масових відкритих онлайн-курсів (МВОК). Він надає широкий вибір безкоштовних курсів, розроблених провідними українськими та світовими університетами, викладачами та фахівцями з різних галузей.

На головній сторінці сайту (рис. 1.1) розміщене велике банерне зображення, яке інформує відвідувачів про нові курси, акції або важливі освітні новини. Це банерне зображення є центральним елементом сторінки, що привертає увагу та сприяє інформуванню користувачів про найактуальніші пропозиції та події. Основне меню, розташоване у верхній частині сторінки, забезпечує швидкий доступ до основних розділів сайту, таких як "Курси", "Корпоративне навчання" та підпискою "Prometheus+".

Розділ "Курси" представляє собою багатий і різноманітний перелік усіх доступних навчальних програм, ретельно категоризованих за тематичними напрямами, такими як література, кіно, мистецтво, музика, журналістика, театр, історія, мода та соціум. Кожен тематичний напрямок представлений яскравими ілюстраціями, що привертають увагу, та короткими описами, що надають ключову інформацію про кожен курс. Це дозволяє користувачам швидко зорієнтуватися у великому обсязі матеріалів і вибрати курс, який найбільше відповідає їхнім

інтересам та потребам. Яскравий дизайн і зручна навігація забезпечують легкий доступ до всієї необхідної інформації, що робить процес вибору курсу доступним.

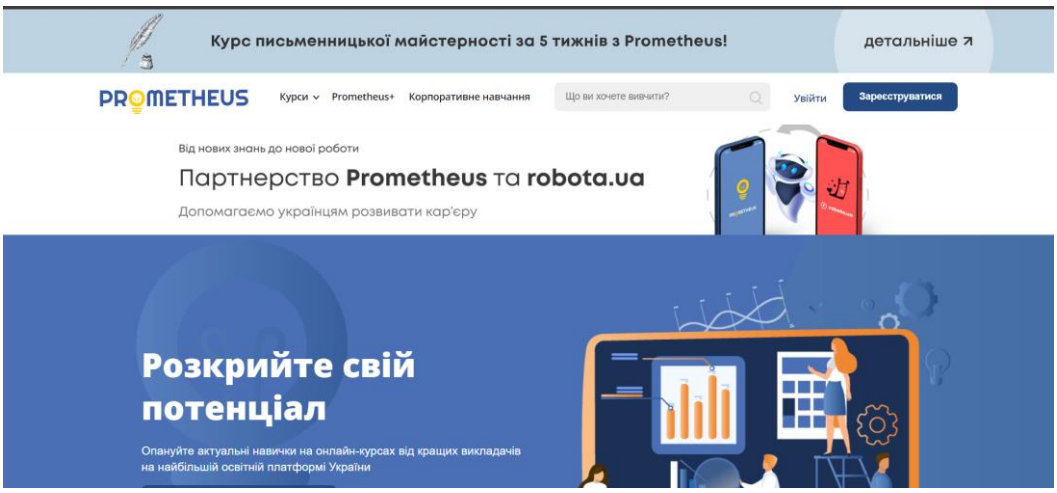


Рисунок 1.1 – Головна сторінка веб-сайту prometheus.org.ua

Сайт пропонує детальну інформацію про кожен курс, включаючи опис, тривалість, розклад, вимоги до слухачів та відгуки. Курси покривають різні теми, такі як IT, бізнес, гуманітарні науки, медицина, освіта тощо. Кожен курс містить відеолекції, інтерактивні завдання, тести та форуми для обговорення.

Процес реєстрації на курс є простим і зручним. Користувачі можуть створити обліковий запис або увійти через соціальні мережі. Після реєстрації користувачі отримують доступ до всіх матеріалів курсу та можуть проходити його у зручному для себе темпі.

Сайт активно залучає користувачів до взаємодії через форуми та групи обговорень, де студенти можуть задавати питання, обговорювати матеріали та ділитися досвідом. Це сприяє створенню спільноти, де учасники можуть підтримувати один одного та покращувати свої знання.

Хоча сайт й є однією з провідних українських платформ для онлайн-освіти, має деякі недоліки, які можуть впливати на зручність і ефективність користування. Порівняно з великими міжнародними платформами, такими як Coursera [9] або

Udemy, Prometheus пропонує меншу кількість курсів, що обмежує вибір для користувачів, які шукають специфічні або менш поширені теми. Крім того, взаємодія між викладачами та студентами може бути недостатньою, особливо коли викладачі не активно підтримують свої курси або не відповідають на питання студентів, що призводить до браку зворотного зв'язку та підтримки для користувачів ресурсу.

Сайт EdEra [11] є веб-порталом української онлайн-платформи для навчання, яка пропонує освітні курси з різних дисциплін. Головна сторінка сайту містить інформацію про переваги платформи та можливості, які вона надає для саморозвитку і підвищення кваліфікації. Дизайн сайту є сучасним і естетично приємним, з використанням яскравих фотографій та інтуїтивно зрозумілого розташування основного меню. На (рис. 1.2) зображено головну сторінку сайту, яка відображає ключові функції та можливості платформи.

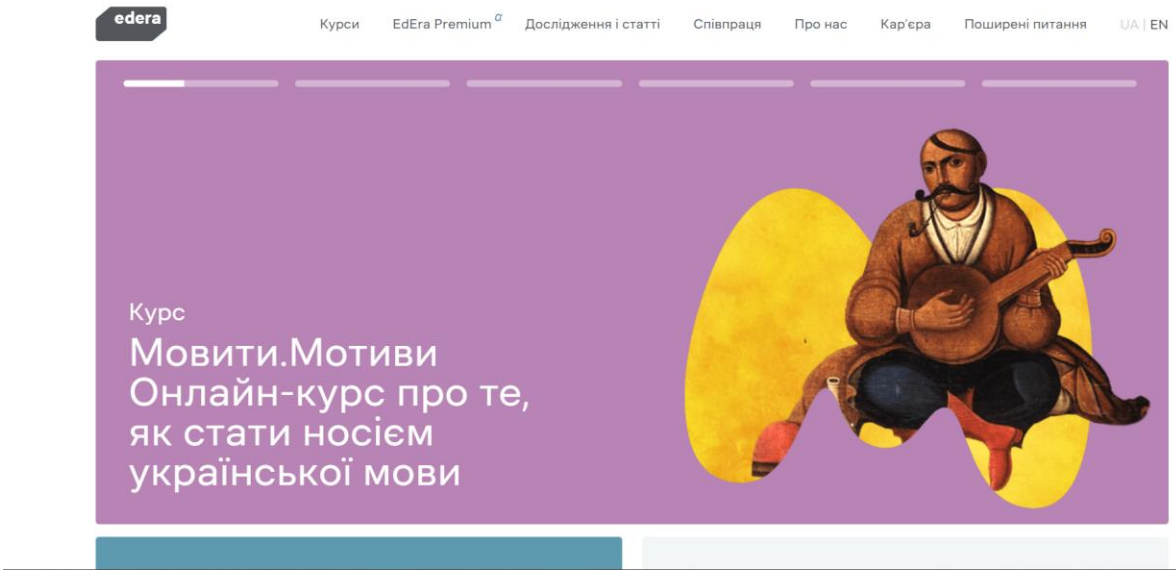


Рисунок 1.2 – Головна сторінка веб-сайту ed-era.com

Сайт пропонує детальну інформацію про різні курси, які доступні на платформі EdEra. Кожен курс має окрему сторінку з описом змісту, навчальних

цілей, ілюстраціями та інформацією про тривалість і вимоги до слухачів. Це дозволяє користувачам ознайомитися з усіма можливостями платформи і вибрати оптимальний курс для своїх потреб.

Окремий розділ сайту присвячений викладачам. Кожен викладач представлений з фотографією, коротким описом кваліфікації та досвідом роботи. Це дозволяє користувачам мати довіру до науково-педагогічного складу та зробити обізнаний вибір курсу.

Сайт також надає інформацію про акції та спеціальні пропозиції, що стимулює користувачів до проходження курсів. Наявність блогу з корисними статтями про освіту також сприяє підвищенню довіри до платформи та підтримує взаємодію з користувачами.

Однак, сайт може бути покращений шляхом додавання функціональності обговорення курсів або спільнот, де користувачі могли б ділитися досвідом та отримувати підтримку. Це забезпечить зручність для користувачів і сприятиме створенню активної освітньої спільноти. Також було б корисно включити розділ з частими запитаннями та детальною контактною інформацією для зв'язку з адміністрацією платформи.

Сайт VUMonline.ua [1] є українською платформою для онлайн-навчання, що надає доступ до різноманітних освітніх курсів. Головна сторінка (рис. 1.3) демонструє основні переваги платформи. На цій сторінці розміщене велике банерне зображення, яке інформує відвідувачів про нові курси, акції або важливі освітні новини. Банер є центральним елементом сторінки, привертаючи увагу користувачів та надаючи їм актуальну інформацію.

Крім банеру, на головній сторінці розміщені блоки з рекомендаціями курсів, які можуть зацікавити користувачів, враховуючи їхні попередні вибори або популярність серед інших студентів. Також, присутній розділ з відгуками

користувачів, який дозволяє новачкам ознайомитися з досвідом тих, хто вже пройшов певні курси, що додатково допомагає у виборі навчальної програми.

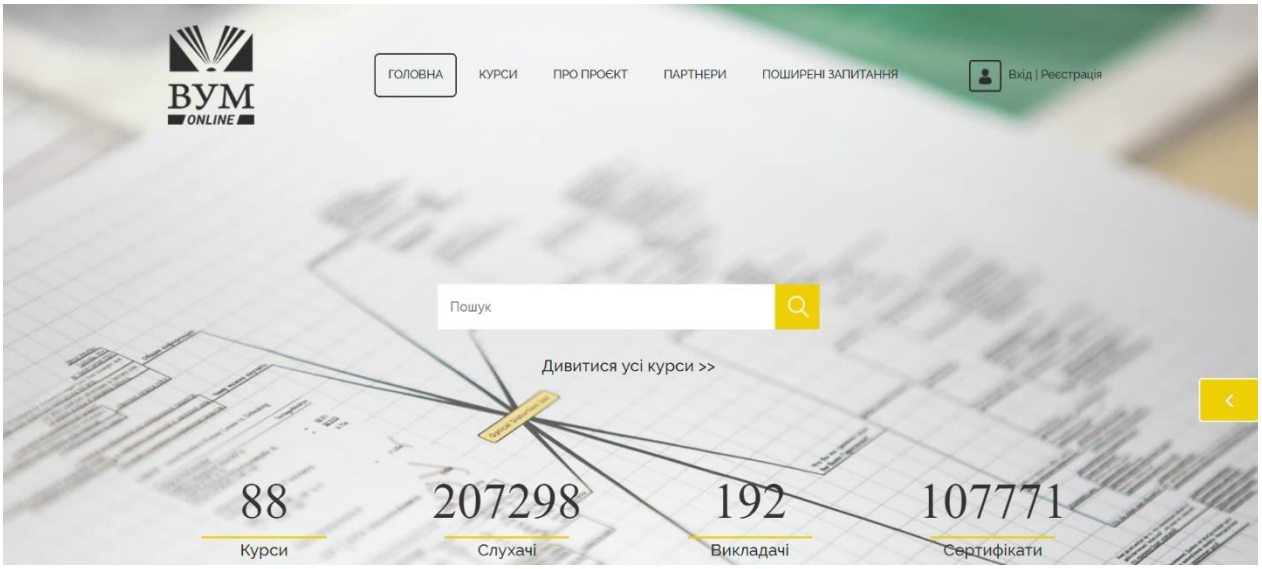


Рисунок 1.3 – Головна сторінка веб-сайту vumonline.ua

Платформа надає детальну інформацію про кожен курс, включаючи опис змісту, навчальні цілі, тривалість та вимоги до слухачів. Це дозволяє користувачам обирати курси, які найбільше відповідають їхнім інтересам і потребам.

Розділ, присвячений викладачам, містить фотографії та короткі біографії кожного викладача, зокрема інформацію про їхню кваліфікацію та професійний досвід. Це допомагає користувачам впевнено обирати курси, знаючи про високий рівень знань та навичок викладання.

Одним з недоліків сайту є недостатня адаптація для мобільних пристроїв. Інтерфейс та функціональність сайту можуть не забезпечувати повної зручності для користувачів, які навчаються з мобільних телефонів або планшетів. Покращення мобільної версії сайту зробило б його більш доступним та привабливим для широкої аудиторії, забезпечуючи комфортний доступ до освітніх ресурсів у будь-якому місці.

Сайт WiseCow.com.ua є українським освітнім ресурсом, який надає доступ до безкоштовних відеолекцій з різних галузей знань. Платформа орієнтована на забезпечення рівного доступу до освітніх матеріалів для всіх бажаючих, незалежно від місця проживання.

Головна сторінка веб-сайту WiseCow.com.ua (рис. 1.4) присвячена курсам і навчальним матеріалам. Вона створена для зручної навігації і надання користувачам швидкого доступу до різноманітних освітніх ресурсів. На головній сторінці розташовані яскраві ілюстрації та короткі описи основних розділів, серед яких література, кіно, мистецтво, музика, журналістика, театр, історія, мода та соціум. Кожен розділ має свої унікальні зображення та інформаційні блоки, що допомагають користувачам швидко знайти необхідні курси та навчальні матеріали.

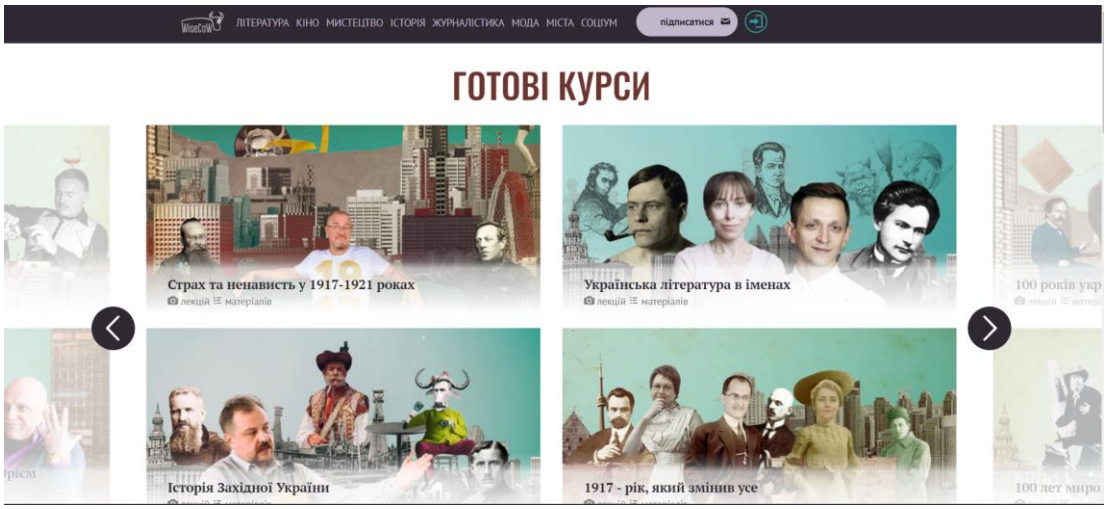


Рисунок 1.4 – Головна сторінка веб-сайту WiseCow.com.ua

Сайт пропонує різноманітні курси, кожен з яких містить 10 відео та додаткові матеріали. Це дозволяє користувачам отримувати знання в зручному для них темпі, переглядаючи лекції у будь-який час.

Основним недоліком сайту є його перезавантаженість інформацією та мультимедійними елементами. Велика кількість зображень, відео та тексту може уповільнювати завантаження сторінок і ускладнювати навігацію, особливо для

користувачів з повільним інтернет-з'єднанням або тих, хто використовує мобільні пристрої. Це може призводити до негативного користувацького досвіду, знижуючи загальну зручність використання платформи загалом.

В цілому, wisecow.com.ua є корисним ресурсом для тих, хто прагне до саморозвитку і хоче мати доступ до якісних освітніх матеріалів. Однак, вирішення проблеми перезавантаженості сайту могло б значно покращити користувацький досвід і зробити платформу ще більш доступною та зручною.

1.2 Порівняльний аналіз розглянутих програм-аналогів

Сайт з можливістю управління освітніми матеріалами та навчальними курсами є значно кращим і більш зручним для користувачів порівняно з традиційними платформами. Такий сайт надає ряд переваг, які полегшують процес навчання та покращують загальний досвід користувачів.

Однією з основних переваг сайту з можливістю управління навчальними матеріалами є зручність для користувачів. Вони можуть легко організовувати та структурувати свої освітні матеріали, створювати курси, додавати завдання та ресурси, не потребуючи додаткових зусиль чи відвідування офісу навчального закладу. Це забезпечує гнучкість та зручність, особливо для тих, хто має обмежений час або перебуває у віддаленому місці. Користувачі можуть з легкістю вибирати підходящі курси, додавати нові матеріали та контролювати процес навчання за декілька кліків.

Крім того, сайт з можливістю управління освітніми матеріалами допомагає економити час і зусилля як для викладачів, так і для студентів. Користувачі можуть швидко додавати та оновлювати навчальні матеріали, не потребуючи взаємодії з технічною підтримкою чи адміністрацією. Це дозволяє уникнути непотрібних затримок і стресу, пов'язаного з організацією навчального процесу. Для навчальних закладів це дозволяє автоматизувати процес управління матеріалами та курсами,

забезпечуючи більш ефективне використання ресурсів та уникнення конфліктів у розкладі занять, лекцій тощо.

Також варто відзначити, що сайт з функціональністю управління освітніми матеріалами покращує загальний користувацький досвід. Користувачі отримують більше контролю над процесом навчання та можуть зручно планувати свій час. Вони можуть вибирати оптимальний для себе час для проходження курсів, залежно від своїх особистих потреб та розкладу. Завдяки можливості попереднього завантаження матеріалів або вказання особливих вимог, користувачі мають можливість підготуватися до занять і максимально ефективно використовувати час.

Крім зручності для користувачів, наявність функції управління освітніми матеріалами сприяє покращенню організаційного процесу в навчальному закладі. Це допомагає уникнути перевантаження адміністративного персоналу, зменшує кількість запитів та покращує управління навчальними курсами. Навчальний заклад отримує більш точну і структуровану інформацію про матеріали та курси, що сприяє плануванню робочого часу викладачів та оптимальному розподілу ресурсів.

Необхідно також зазначити, що наявність онлайн-управління освітніми матеріалами є показником сучасності та інноваційності навчального закладу. Це демонструє прагнення до використання сучасних технологій для поліпшення обслуговування користувачів і надання їм максимального комфорту. Такий підхід сприяє побудові довгострокових взаємовигідних відносин з користувачами і створює позитивний імідж навчального закладу.

Однак, один із недоліків такого сайту може бути перезавантаженість інформацією. Велика кількість матеріалів та курсів може ускладнювати навігацію та пошук необхідної інформації. Це може призводити до плутанини і знижувати ефективність використання платформи. Вирішення цієї проблеми шляхом оптимізації інтерфейсу та структури сайту могло б значно покращити користувацький досвід, забезпечити кращу навігацію та зробити платформу більш зручною та ефективною для всіх користувачів.

1.3 Постановка задачі

Для створення веб-сайту для управління освітніми матеріалами та навчальними курсами за допомогою технологій TypeScript[10], Tailwind CSS [18], HTML, Node.js [16] та Next.js [15] були виконані наступні кроки:

– було проведено детальний аналіз існуючих платформ для управління освітніми ресурсами, таких як Prometheus, ВУМ online, EdEra та WiseCow, щоб визначити їхні переваги та недоліки;

– було визначено основні функції та можливості, які мають бути реалізовані на веб-сайті;

– було обрано основні технології розробки, такі як TypeScript для програмування, Node.js для бекенду, та Next.js для фронтенду;

– було розроблено зручний та сучасний дизайн для веб-сайту, використовуючи HTML та Tailwind CSS для створення інтуїтивного користувацького інтерфейсу;

– було сплановано структуру сторінок та компоненти, які дозволяють користувачам легко керувати освітніми матеріалами та курсами;

– було створено бекенд для веб-сайту, використовуючи Node.js; було розроблено API, яке обробляє запити від користувачів, такі як додавання, редагування та видалення освітніх матеріалів та курсів;

– було забезпечено надійну взаємодію з базою даних для зберігання інформації користувачів;

– було реалізовано функціонал, що дозволяє користувачам додавати нові курси, завантажувати освітні матеріали та керувати ними;

– було забезпечено надійні механізми автентифікації та авторизації для захисту доступу до функцій веб-сайту;

– було використано сучасні методи шифрування та захисту даних, щоб гарантувати безпеку користувачів;

– було спроектовано та створено структуру бази даних для зберігання інформації про курси, матеріали, користувачів та інші необхідні дані для системи;

– було забезпечено ефективну взаємодію з базою даних за допомогою відповідних технологій.

Висновки до розділу 1

У цьому розділі було проведено детальний огляд існуючих українських освітніх платформ, таких як Prometheus, ВУМ online, EdEra і WiseCow. Кожна з цих платформ має свої унікальні функції та переваги, які роблять їх привабливими для різних категорій користувачів.

Prometheus є однією з найвідоміших платформ, яка пропонує безкоштовні онлайн-курси від провідних університетів та організацій. Основною перевагою Prometheus є широкий вибір курсів різної тематики, висока якість викладання та доступність для широкої аудиторії. Однак, платформа стикається з викликами у забезпеченні стабільності роботи під час пікових навантажень та адаптації курсів для користувачів з різними освітніми потребами користувачів.

ВУМ online (VUMonline.ua) відрізняється своїм акцентом на інноваційних методах навчання та інтеграції сучасних технологій у навчальний процес. Платформа надає курси з різних галузей знань, зокрема літератури, кіно, мистецтва, музики, журналістики, театру, історії, моди та соціальних наук. Головна сторінка сайту демонструє основні переваги платформи, такі як зручність онлайн-навчання, високий рівень викладачів та сучасні методики викладання. Проте ВУМ online також стикається з проблемами забезпечення стабільності роботи та інтеграції з мультимедійними ресурсами для підвищення користувацького досвіду.

EdEra виділяється своїм фокусом на інтерактивність та використанням мультимедійних матеріалів у навчанні. Платформа пропонує курси, що включають відеолекції, інтерактивні завдання та форуми для обговорення, що сприяє

глибокому зануренню у навчальний матеріал. Незважаючи на високий рівень інтерактивності, EdEra потребує подальшого розвитку в напрямку забезпечення доступності для користувачів з особливими освітніми потребами та покращення стабільності роботи під час високих навантажень.

WiseCow відома своїм багатим вибором курсів, що охоплюють різні сфери знань, та зручним інтерфейсом для користувачів. Платформа акцентує увагу на якості освітніх матеріалів та зручності навігації. Одним з головних викликів для WiseCow є забезпечення стабільної роботи та інтеграції з іншими освітніми ресурсами, щоб забезпечити безперервність навчального процесу.

Під час аналізу було визначено спільні виклики, з якими стикаються всі ці платформи, включаючи адаптацію до різних освітніх потреб, інтеграцію з мультимедійними ресурсами, забезпечення стабільності та високої продуктивності під час пікових навчальних періодів, та реалізацію потреб інклюзивності.

На основі цього аналізу було окреслено напрямки для подальшого розвитку платформ управління освітніми матеріалами. Перш за все, необхідно розробити більш гнучкі і налаштовувані інструменти, що дозволять користувачам адаптувати навчальний процес під свої індивідуальні потреби. Підвищення інтегрованості та інтерактивності освітніх ресурсів сприятиме більш глибокому залученню студентів у навчальний процес та покращенню їхніх результатів. Крім того, покращення стабільності та швидкості роботи платформ є критичним для забезпечення безперервного та ефективного навчання дистанційно.

Отже, перехід до розробки і впровадження більш передових технологій у сфері управління освітніми матеріалами відкриває шлях для створення більш ефективних, гнучких та доступних навчальних середовищ, що може значно покращити якість освіти. Визначені напрямки розвитку та розробки нових інструментів та підходів стануть ключем до вирішення існуючих проблем та сприятимуть підвищенню ефективності онлайн-навчання в Україні та у світі.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ОСВІТНІМИ МАТЕРІАЛАМИ

2.1 Архітектура та технологічна реалізація системи керування освітніми матеріалами

Для управління освітніми матеріалами буде створено систему, що використовує сучасні технології такі як TypeScript [10], Node.js [6], Next.js [7] та MongoDB [8], Redux [9]. Цей підхід забезпечить ефективне управління матеріалами та високу доступність і зручність для студентів і викладачів.

Основними компонентами реалізації будуть:

- серверна частина системи буде реалізована на мові програмування TypeScript, яка додає строгу типізацію та вдосконалені можливості до JavaScript, забезпечуючи більшу безпеку та легкість у підтримці коду;
- використання Node.js дозволить створити ефективний та швидкий бекенд, що зможе обробляти велику кількість запитів паралельно завдяки неблокуючій архітектурі;
- MongoDB, як документо-орієнтована база даних, ідеально підходить для гнучкої роботи з великими обсягами структурованих і неструктурованих даних, забезпечуючи швидкий доступ і легке масштабування системи;
- використання Next.js на фронтенді забезпечить створення оптимізованих веб-сторінок з підтримкою серверного рендерингу для швидкого завантаження та відмінної SEO-оптимізації;
- для управління станом додатків на клієнтській стороні використовується Redux. Redux допомагає утримувати стан додатку консистентним і прогнозованим, навіть у складних системах з великою кількістю даних і користувачів;

– застосування Redux забезпечує централізоване управління станом, що полегшує передачу даних між компонентами і спрощує логіку управління станом; це особливо корисно при розробці великих додатків, де необхідно забезпечити швидкий доступ до даних і їх оновлення в реальному часі;

– використання Redux разом з Next.js та TypeScript дозволяє створювати ефективні та надійні рішення для управління складними даними і взаємодіями в додатку, сприяючи вищій продуктивності та кращому користувацькому досвіду у розробленій системі.

Використання зазначених компонентів та технологій дозволить розробити надійну та ефективну систему керування освітніми матеріалами, яка буде відповідати сучасним вимогам до швидкості, безпеки та зручності у розробці.

2.2 TypeScript та його використання у розробці серверної частини

TypeScript – це сучасна, об'єктно-орієнтована мова програмування, розроблена Microsoft, яка додає строгу типізацію та різні інші поліпшення до JavaScript. Ця мова значно підвищує продуктивність, безпеку та масштабованість при розробці різноманітних додатків, включаючи серверні рішення. У цьому розділі ми детально розглянемо ключові особливості TypeScript, його застосування у розробці серверної частини, а також обґрунтуємо вибір цієї мови для проекту.

TypeScript компілюється в JavaScript, що дозволяє виконувати код на будь-якому JavaScript-двигуні з високою продуктивністю. Компіляція TypeScript в JavaScript перед виконанням забезпечує додаткову перевірку типів і виявлення помилок на ранніх етапах розробки, що є надзвичайно важливим для серверних додатків, де надійність та стабільність мають критичне значення. На рис (2.1) зображено схему виконання коду TypeScript.

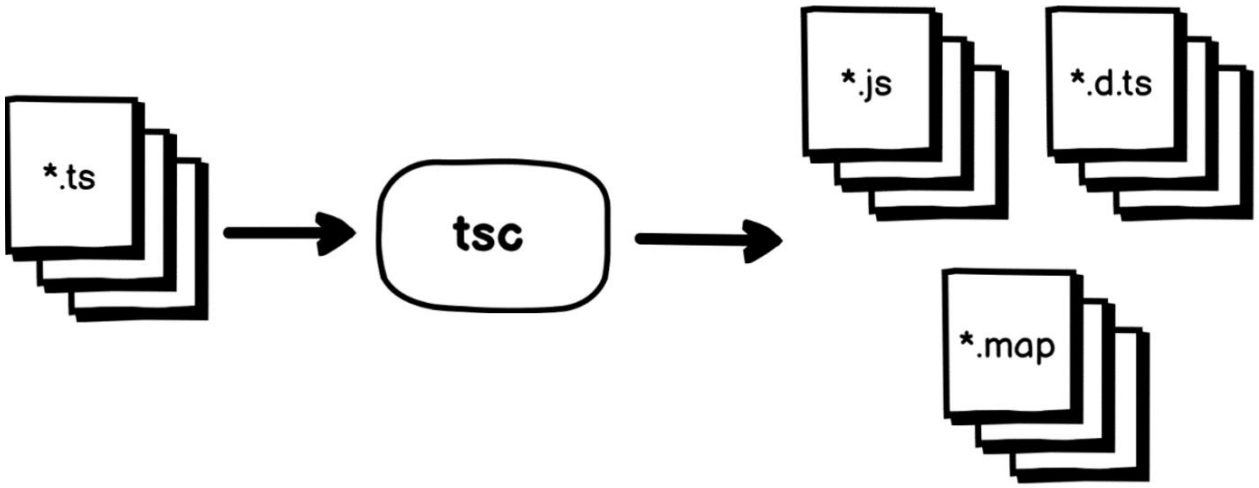


Рисунок 2.1 – Схема компілятора TypeScript, який перетворює код TypeScript в JavaScript, а також деякі файли визначення типів або вихідні карти

Сфери використання: TypeScript широко застосовується для створення веб-застосунків, десктопних програм, мобільних додатків, а також у розробці серверних рішень з використанням Node.js. Завдяки інтеграції з сучасними фреймворками та бібліотеками, як-от Angular, React і Vue.js, TypeScript дозволяє створювати масштабовані та безпечні веб-додатки.

Обґрунтування вибору: вибір TypeScript для розробки серверної частини проекту обумовлений кількома ключовими аспектами. По-перше, строга типізація та об'єктно-орієнтовані можливості мови значно підвищують якість коду та спрощують його підтримку та розширення. По-друге, велика сумісність з JavaScript екосистемою дозволяє легко інтегрувати різноманітні інструменти та бібліотеки, що підвищує продуктивність розробки. Нарешті, TypeScript підтримує сучасні програмні парадигми, такі як асинхронне програмування та модулі, що є важливим для ефективної обробки великих обсягів запитів та даних. Ці характеристики роблять TypeScript ідеальним вибором для розробки серверної частини нашого проекту, де необхідні висока продуктивність, надійність та масштабованість.

Ці аспекти роблять TypeScript ідеальним вибором для розробки серверної частини нашого проекту, де потрібна висока продуктивність та масштабованість.

2.3 Архітектура серверної частини

Архітектура серверної частини системи керування освітніми матеріалами заснована на використанні паттерну Controller-Service-Repository. Цей підхід сприяє ясному розподілу відповідальностей та дотриманню принципів SOLID, забезпечуючи легкість в розробці, тестуванні та підтримці програмного продукту. На рисунку 2.2 демонструється структура цього паттерну для проекту.

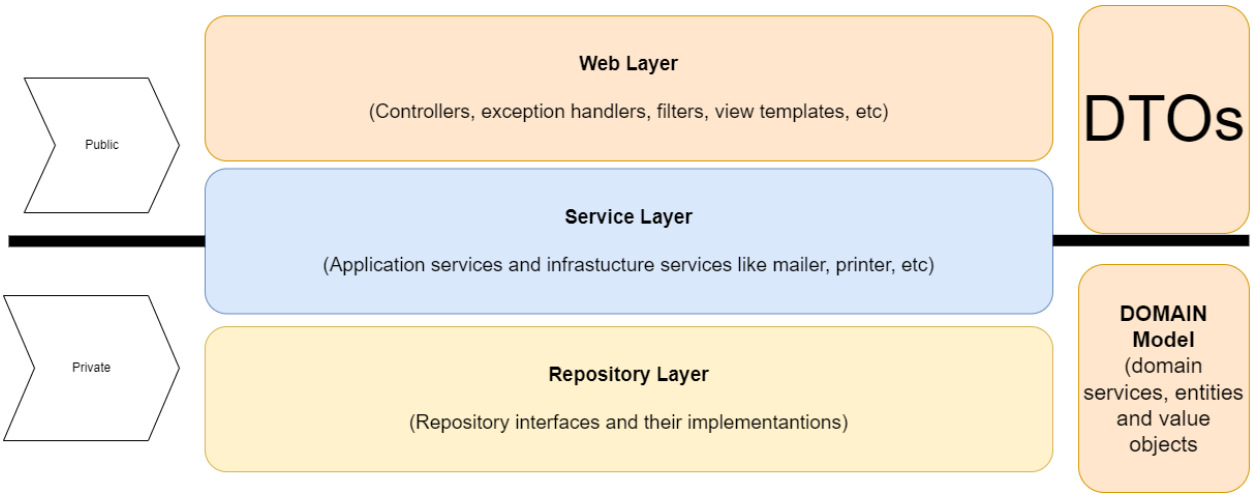


Рисунок 2.2 – Паттерн Controller-Service-Repository

Контролери (Controllers): контролери відіграють ключову роль у обробці вхідних запитів сервера. Вони аналізують, валідують вхідні дані і направляють їх до відповідних сервісів для реалізації бізнес-логіки та обробки помилок.

Сервіси (Services): сервіси утілюють бізнес-логіку системи та здійснюють маніпуляції з даними. Вони приймають запити від контролерів і виконують необхідні операції, такі як взаємодія з репозиторіями для отримання, зміни даних або інші специфічні для домену дії.

Репозиторії (Repositories): репозиторії забезпечують доступ до різних джерел даних, як-от бази даних чи зовнішні сервіси. Вони ізолюють логіку доступу до даних від сервісів та контролерів, що дозволяє змінювати джерела даних без впливу

на решту системи. Репозиторії також можуть включати додаткові функції, які покращують продуктивність, наприклад, кешування.

Така структура архітектури гарантує гнучкість, модульність і масштабованість серверної частини проекту, що є важливим для ефективного розвитку, тестування і підтримки системи управління освітніми матеріалами.

2.4 Node.js та його використання у розробці серверної частини

Node.js є потужним середовищем виконання, що базується на JavaScript-двигуні V8 від Google, яке дозволяє розробникам використовувати JavaScript для написання серверного коду. Використання TypeScript з Node.js подальше підвищує ефективність та надійність розробки завдяки статичній типізації та розширеним можливостям для організації коду, що особливо корисно в великих проектах.

Основні переваги Node.js з TypeScript:

– статична типізація та компіляція: TypeScript забезпечує статичну типізацію, що допомагає уникнути багатьох помилок під час компіляції, замість виконання на продакшені. Це підвищує надійність коду та полегшує масштабування додатків;

– підвищена продуктивність розробки: TypeScript додає додаткові функціональні можливості, такі як перевантаження функцій, enums, інтерфейси, які роблять розробку більш інтуїтивною та безпечною. Інтеграція з популярними IDE, такими як Visual Studio Code, забезпечує відмінну підтримку з автодоповненням і рефакторингом коду;

– асинхронна обробка: Node.js використовує асинхронні події та неблокуючий ввод-вивід, що дозволяє обробляти велику кількість запитів одночасно без затримок. TypeScript покращує роботу з асинхронними і проміс-орієнтованими патернами, роблячи код більш чистим та зрозумілим;

– широка підтримка баз даних: з Node.js легко інтегрувати різні бази даних, включаючи NoSQL (наприклад, MongoDB), реляційні (наприклад, PostgreSQL, MySQL) та інші популярні системи через різноманітні драйвери та ORM. TypeScript покращує роботу з ORM системами, забезпечуючи сильне типізування та об'єктно-орієнтоване проєктування;

– спільнота та екосистема: широка спільнота Node.js та бібліотеки, доступні через npm, забезпечують потужні інструменти та фреймворки, які можуть бути легко адаптовані під TypeScript для розширення можливостей розробки.

У контексті систем управління освітніми матеріалами, використання Node.js та TypeScript може виявитися надзвичайно корисним для розробки серверної частини, забезпечуючи високу продуктивність, масштабованість та гнучкість розширення та підтримки великих систем проекту. Завдяки цим технологіям розробники можуть легко інтегрувати різноманітні функції, забезпечуючи швидку обробку даних та оптимізовану роботу серверу.

2.5 Обґрунтування використання MongoDB у системі керування освітніми матеріалами та навчальними курсами

MongoDB, яка є провідною NoSQL базою даних, вирізняється своєю гнучкістю структур даних і орієнтацією на високу продуктивність при роботі з великими обсягами даних. Це відкрита система управління базами даних, що забезпечує високий рівень масштабованості і доступності.

Обґрунтування вибору MongoDB:

– гнучка схема даних: MongoDB дозволяє зберігати документи в форматі BSON (бінарний JSON), що сприяє гнучкості управління різними видами освітніх матеріалів. Така модель добре підходить для систем, де необхідно часто змінювати структуру даних;

– масштабованість: MongoDB підтримує горизонтальне масштабування через шардінг, дозволяючи системі ефективно обробляти зростаючий обсяг даних і користувачів без значної втрати продуктивності;

– швидка обробка великих обсягів даних: використання індексації та вбудованих операцій з агрегації дозволяє MongoDB швидко обробляти великі масиви даних, що критично для аналітики та звітності в освітніх системах;

– підтримка різноманітних типів запитів: MongoDB може ефективно обробляти запити на пошук, агрегацію, та текстильний пошук, що дозволяє більш гнучко взаємодіяти з даними і спрощує пошук та організацію освітніх ресурсів;

– висока доступність: реплікаційні набори у MongoDB забезпечують автоматичне створення копій даних та високу доступність, мінімізуючи ризик втрати даних у разі збою системи.

Вибір MongoDB відповідає вимогам проекту, оскільки:

– гнучкість управління даними сприяє легкій інтеграції нових видів освітніх матеріалів і форматів, забезпечуючи можливість швидко адаптувати систему під змінювані потреби користувачів.

– масштабованість забезпечує стабільність роботи системи при збільшенні обсягів даних та користувачів.

– швидка обробка великих обсягів даних важлива для аналітики та звітності, що може використовуватися для оцінки ефективності навчальних програм і матеріалів.

Обрання MongoDB як основи для системи керування освітніми матеріалами та навчальними курсами забезпечує стабільну, надійну та легко адаптовану платформу для управління різноманітними освітніми ресурсами.

2.7 Tailwind CSS

Tailwind CSS є сучасною утилітарною CSS-фреймворком, що дозволяє розробникам швидко створювати налаштовані дизайни без потреби писати багато власного CSS. У проекті управління освітніми матеріалами та навчальними курсами, Tailwind CSS використовується для ефективної роботи з інтерфейсом, забезпечуючи адаптивність, консистентність та привабливість веб-сайту.

Основні переваги Tailwind CSS:

- Tailwind CSS дозволяє легко створювати адаптивні макети за допомогою вбудованих утиліт для респонсивного дизайну;
- веб-сайт автоматично адаптується до різних розмірів екранів та пристроїв, забезпечуючи користувачам оптимальний досвід перегляду;
- завдяки утилітарному підходу, Tailwind CSS дозволяє розробникам використовувати лише ті стилі, які необхідні для компонентів у системі;
- Tailwind CSS спрощує процес створення інтерфейсу, дозволяючи розробникам швидко застосовувати стилі без необхідності писати складні CSS правила для цього;
- конфігурація Tailwind CSS сприяє уніфікації стилів на всьому веб-сайті для управління освітніми матеріалами;
- розробники можуть визначати теми, кольори та шрифти в одному місці, що гарантує консистентність візуального дизайну;
- Tailwind CSS легко інтегрується з React, роблячи його ідеальним вибором для проектів, які використовують ці технології.

Використання Tailwind CSS у цьому проекті не тільки покращило візуальний аспект веб-сайту, але й сприяло кращій організації та підтримці стилів, роблячи процес розробки більш ефективним та гнучким.

2.8 Redux

Redux є популярною бібліотекою для управління станом додатків, яка широко використовується у розробці складних інтерфейсів користувача на React. Ця бібліотека допомагає розробникам створювати додатки, які поводяться послідовно, працюють у різних середовищах (клієнт, сервер, додатки) і легко тестуються.

У контексті системи управління освітніми матеріалами, Redux використовується для створення чіткої і передбачуваної моделі поведінки додатку, що полегшує управління станами додатку, зокрема даними користувачів, списками курсів, інформацією про лекції та користувацькі сесії. Це забезпечує однорідний потік даних і дозволяє розробникам краще контролювати поведінку додатку через єдине джерело істини (Single Source of Truth), що зменшує ймовірність помилок та забезпечує цілісність даних.

2.9 Проектування архітектури серверної частини

У цьому розділі розглянуто детальну архітектуру серверної частини проекту, яка базується на паттерні Controller-Service-Repository.

Контролери служать в якості в'язки між користувачем і логікою програми. Вони обробляють вхідні запити від користувачів і відправляють відповіді. Ось декілька прикладів контролерів:

- контролер курсів (CourseController): управління курсами, включно з додаванням, оновленням та видаленням інформації про курси;
- контролер користувачів (UserController): управління профілями користувачів, включаючи аутентифікацію та авторизацію;
- контролер макетів (LayoutController): керування налаштуваннями макету інтерфейсу користувачів.

Сервіси забезпечують бізнес-логіку та операції над даними, які використовують контролери. Вони служать для абстракції та повторного використання коду. Сервіси, які є в проєкті:

- сервіс курсів (CourseService): відповідає за усю бізнес логіку, що пов'язана з курсами там взаємодію з ними;
- сервіс користувачів (UserService): управління даними користувачів, включаючи реєстрацію, логін та управління профілями.

Репозиторії відповідають за взаємодію з базою даних. Вони дозволяють ізолювати логіку доступу до даних від бізнес-логіки, що спрощує модифікації і тестування. Репозиторії, які є в проєкті

- репозиторій курсів (CourseModel): працює з базою даних для зберігання та витягування даних про курси;
- репозиторій користувачів (UserModel): зберігання даних користувачів, робота з авторизацією;
- репозиторій макетів (LayoutModel): управління даними макету.

Кожен контролер взаємодіє зі своїм відповідним сервісом для виконання конкретних функцій. Це дозволяє забезпечити ефективну та структуровану обробку даних у серверній частині застосунку.

Висновки до розділу 2

Розділ 2 детально описує ключові аспекти проектування та розробки системи керування освітніми матеріалами. На основі цього аналізу можна зробити висновки:

Використання TypeScript, Node.js, для серверної частини гарантує високу продуктивність, безпеку та гнучкість системи. Ці технології дозволяють ефективно управляти запитами та даними, забезпечуючи стабільність та надійність системи.

Застосування паттерну Controller-Service-Repository сприяє ясному розподілу відповідальностей, що забезпечує легкість в розробці, тестуванні та підтримці

програмного продукту. Ця архітектура допомагає в організації коду та забезпечує масштабованість при розвитку системи.

Обрання MongoDB як СУБД відповідає потребам системи за гнучкістю, масштабованістю та продуктивністю. Документо-орієнтована структура дозволяє зберігати і обробляти різноманітні типи даних, що робить її ідеальною для освітніх систем, які працюють з великим обсягом як структурованих, так і неструктурованих даних. Висока продуктивність і можливість горизонтального масштабування забезпечують стабільність та надійність навіть при інтенсивному використанні.

Застосування Next.JS для розробки клієнтської частини забезпечує динамічність та відгуковість інтерфейсу, що покращує користувацький досвід. Модульний підхід та віртуальний DOM дозволяють підтримувати високу продуктивність інтерфейсу навіть при інтенсивному використанні.

Вибір Redux для управління станом додатку забезпечує передбачуваність і контроль у взаємодії компонентів React. Використання Redux дозволяє централізовано зберігати стан додатку, що полегшує керування даними та їх оновлення. Це сприяє зменшенню складності розробки великих додатків, оскільки стан управління чітко визначений і легко відстежуваний. Крім того, Redux забезпечує високу інтеграцію з іншими інструментами і бібліотеками, що дозволяє ефективніше використовувати ресурси та підвищує продуктивність розробки.

Ці аспекти створюють міцну основу для розробки системи керування освітніми матеріалами, що не лише відповідає сучасним вимогам до швидкості та безпеки, але й надає гнучкість.

РОЗДІЛ 3. РОЗРОБКА СЕРВЕРНОЇ ТА КЛІЄНСТЬКОЇ ЧАСТИНИ, ЇХ РОЗМІЩЕННЯ В ІНТЕРНЕТ

У цьому розділі описано процес розробки серверної частини програмного забезпечення, спрямованого на управління освітніми курсами, їхніми матеріалами, а також інтеракції з користувачами. Розробка розділена на три основні частини: розробка серверної, клієнтської частини додатку та розміщення для загального доступу в мережу “Інтернет”.

3.1 Розробка серверної частини

У цьому підрозділі описано процес розробки серверної частини програмного забезпечення, яка відповідає за зберігання та створення даних. Надається детальний опис реалізації ключових компонентів серверної частини, включаючи API для взаємодії з клієнтською частиною, а також реалізацію сервісів, які обробляють бізнес-логіку. Опис містить структуровану інформацію про побудову контролерів, сервісів та репозиторіїв.

3.1.1 Розробка моделей

Для зберігання даних у базі даних MongoDB[9] використовується бібліотека mongoose. Нижче наведено опис схем та їх взаємозв'язків:

– userSchema: визначає схему користувача з полями для імені, електронної пошти, паролю, аватара, ролі, статусу верифікації та курсів;

– courseSchema: визначає схему даних курсу з полями для URL відео, мініатюри відео, заголовку, секції відео, опису, тривалості відео, відеоплеєра, а також інших метаданих, які можуть бути необхідними для опису відеоконтенту.

– layoutSchema: визначає схему макету з полями для типу макету, списку часто задаваних питань (FAQ), категорій.

Код реалізації описаних сервісів наведено у Додатку А.

3.1.2 Використання Next.js

React Next.js є потужним фреймворком для розробки серверно-рендерених або статично генерованих веб-сайтів і додатків, побудованим на основі React. Розроблений компанією Vercel, Next.js спрощує процес розробки, автоматизує багато складних аспектів, таких як маршрутизація сторінок і оптимізація вивантаження, і дозволяє легко втілювати оптимізовані для пошукових систем (SEO) рішення.

Обґрунтування вибору Next.js:

– Next.js використовує файлову систему для маршрутизації, де сторінки автоматично визначаються відповідно до файлів у папці pages. Це робить структуру проєкту чіткою і простою для розуміння системи;

– Next.js дозволяє вибрати між серверним рендерингом і статичною генерацією для кожної сторінки окремо, забезпечуючи гнучкість для оптимізації продуктивності та SEO;

– Next.js автоматично оптимізує додатки, використовуючи такі функції, як автоматичне розділення коду, що дозволяє завантажувати лише необхідні модулі для кожної сторінки;

– розробники можуть легко включати стилі на рівні компонентів за допомогою CSS-модулів або Sass, що спрощує стилізацію і забезпечує ізоляцію стилів та полегшує підтримку та розвиток інтерфейсу.

– Next.js дозволяє створювати API ендпойнти безпосередньо в проєкті, що робить можливим використання серверної логіки та баз даних без додаткового серверного коду або конфігурації;

– широка підтримка від спільноти і постійно оновлюваний набір інструментів забезпечують, що Next.js залишається в лідерах сучасних технологій.

Next.js ідеально підходить для проєктів, де потрібна швидка взаємодія з користувачем і висока продуктивність серверного рендерингу, зокрема для комплексних комерційних додатків і платформ, таких як системи керування освітніми матеріалами. Ці особливості роблять Next.js потужним інструментом для розробки інтерфейсів, що вимагають високої продуктивності та оптимізації для пошукових систем.

3.1.3 Розробка сервісів

Сервіси у серверній частині Node.js — це компоненти, які інкапсулюють бізнес-логіку та забезпечують виконання основних операцій додатку. Вони виконують роль посередників між контролерами і моделями, допомагаючи розділити відповідальність та організувати код у модульний і зрозумілий спосіб.

CourseService: визначає два сервіси для управління курсами в системі. Короткий опис кожної функції:

– використовується для створення нового курсу;

– приймає дані курсу, зберігає їх у базі даних за допомогою моделі CourseModel та повертає успішну відповідь з інформацією про створений курс;

– використовується для отримання всіх курсів;

– витягує всі курси з бази даних, сортує їх за датою створення в порядку спадання і повертає успішну відповідь з переліком курсів.

– UserService: визначає три сервіси для управління користувачами в системі. Він використовує і Redis для кешування даних про користувачів, а також обробляє запити за допомогою Express. Короткий опис кожної функції:

– getUserById: використовується для отримання користувача за його ідентифікатором (id), включаючи всю доступну інформацію про користувача;

– getAllUsersService: використовується для отримання всіх користувачів з бази даних, включаючи їхню основну інформацію;

– updateUserRoleService: використовується для оновлення ролі користувача в системі, забезпечуючи можливість зміни його прав доступу.

Код реалізації описаних сервісів наведено у Додатку Б.

3.1.4 Розробка контролерів

Контролер — це ключовий компонент серверної частини додатка, який відповідає за обробку HTTP-запитів та взаємодію з іншими частинами системи. Контролери виконують бізнес-логіку додатка і визначають, як обробляти дані, що надходять від клієнтів, а також які відповіді відправляти назад.

Контролер користувачів реалізує основні функції для управління користувачами, такі як реєстрація, аутентифікація, активація облікового запису, оновлення профілю, зміна паролю, а також адміністрування (отримання всіх користувачів, оновлення ролі, видалення користувача). Основні його функцій:

– реєстрація користувача (registrationUser);
– активація користувача (activateUser);
– аутентифікація користувача (loginUser);
– вихід з системи (logoutUser);
– оновлення токена доступу (updateAccessToken);
– отримання інформації про користувача (getUserInfo);
– оновлення інформації про користувача (updateUserInfo);
– зміна паролю користувача (updatePassword);
– оновлення аватару користувача (updateProfilePicture);
– отримання всіх користувачів (getAllUsers);
– оновлення ролі користувача (updateUserRole);
– видалення користувача (deleteUser).

Контролер курсів відповідає за обробку різних запитів, пов'язаних з управлінням курсами, такими як завантаження, редагування, перегляд та видалення курсів, а також взаємодію користувачів з курсами, включаючи додавання питань, відповідей, оглядів та коментарів до оглядів. Опис його основних функцій:

– завантаження курсу (uploadCourse);
– редагування курсу (editCourse);
– отримання одного курсу (getSingleCourse);
– отримання всіх курсів (getAllCourses);
– отримання курсу користувачем (getCourseByUser);
– додавання питання до курсу (addQuestion);
– додавання відповіді до питання курсу (addAnswer).

Код реалізації описаних сервісів наведено у Додатку В.

3.1.5 Розробка utils, middleware

Утиліти у Node.js — це допоміжні функції та модулі, які спрощують виконання різних завдань у додатку. Вони можуть включати функції для роботи з файлами, обробки даних, роботи з мережею, логування, і багато іншого. Використання утиліт дозволяє зменшити кількість повторюваного коду і підвищити ефективність розробки. Утиліти для серверної частини:

– analytics.generator: містить функції для генерації аналітичних даних або звітів; ця утиліта може використовуватися для обробки та аналізу даних з різних джерел, надаючи корисні інсайти для користувачів;
– db: відповідає за налаштування і взаємодію з базою даних; ця утиліта містить функції для підключення до бази даних, виконання запитів, а також управління з'єднаннями з базою даних;

- ErrorHandler: реалізує логіку обробки помилок у додатку; цей файл містить код для створення уніфікованих помилок, які можуть бути передані клієнту, забезпечуючи кращу обробку і відображення помилок у додатку;
- jwt: містить функції для роботи з JSON Web Tokens (JWT); ця утиліта включає створення, перевірку та оновлення токенів доступу та рефреш токенів, забезпечуючи безпечне управління сесіями користувачів;
- redis: включає функції для взаємодії з Redis; цей файл містить код для підключення до Redis, зберігання і отримання даних, а також управління кешем, що допомагає підвищити продуктивність додатку;
- sendMail: містить функції для відправки електронних листів користувачам; ця утиліта забезпечує інтеграцію з поштовими сервісами, дозволяючи легко відправляти листи з різноманітними сповіщеннями та інформацією від системи.

Middleware у Node.js — це функції, які виконуються під час обробки HTTP-запитів у додатку. Вони можуть обробляти запити на різних етапах їх обробки, включаючи аутентифікацію, валідацію, логування, роботу з сесіями та інше. Проміжне програмне забезпечення (Middleware) для серверної частини проекту:

- CatchAsyncErrors: це проміжне програмне забезпечення обертає асинхронні функції та автоматично передає помилки до глобального обробника помилок, забезпечуючи чистоту та зручність коду; воно допомагає уникнути дублювання коду для обробки помилок у асинхронних операціях;
- Authentication Middleware: це проміжне програмне забезпечення перевіряє, чи є користувач аутентифікованим перед тим, як дозволити доступ до певних маршрутів; воно декодує токен доступу, перевіряє його валідність та додає інформацію про користувача до запиту; забезпечує безпеку, обмежуючи доступ до захищених ресурсів лише аутентифікованим користувачам системи;
- Authorization Middleware: це проміжне програмне забезпечення перевіряє, чи має користувач відповідні права доступу для виконання певних дій;

використовується для обмеження доступу до адміністративних маршрутів або функцій, які потребують спеціальних прав доступу; забезпечує контроль доступу та безпеку додатку.

3.1.6 Тестування серверної частини

Postman є одним з найпопулярніших інструментів для тестування API (Application Programming Interface). Це потужний інструмент, який дозволяє розробникам ефективно працювати з API на всіх етапах його розробки, від створення до тестування і документування. Використання Postman має кілька переваг над іншими подібними програмами, що робить його вибором багатьох команд розробників по всьому світу.

Postman [10] — це багатофункціональний інструмент для розробки API, який дозволяє користувачам створювати, тестувати, документувати та моніторити API. Він підтримує всі типи запитів HTTP, такі як GET, POST, PUT, DELETE, PATCH та інші. Користувачі можуть створювати запити з різними параметрами, заголовками та тілам, що робить Postman універсальним інструментом для роботи з API який спрощує багато аспектів роботи з REST API та значною мірою підвищує продуктивність роботи розробників.

Основні переваги роботи Postman:

– автоматизація тестування: можливість автоматизації тестування через скрипти та інтеграцію з CI/CD пайплайнами;

– інтуїтивно зрозумілий інтерфейс: простий у використанні графічний інтерфейс, який полегшує створення і відправлення запитів;

– підтримка колекцій: можливість створювати колекції запитів для організації і повторного використання;

– документування API: автоматичне створення документації для API, що полегшує процес обміну інформацією між командами;

– інтеграції: підтримка інтеграцій з різними інструментами для розробки і моніторингу, такими як Jenkins, GitHub, Slack та інші.

Колекція CapyLearmo у Postman містить декілька папок, кожна з яких відповідає за певний аспект API. Нижче наведено короткий опис основних папок для тестування "User", "Course", "Layout" та "Analytics". На рисунку 3.1 відображено інтерфейс Postman з колекцією CapyLearmo для тестування серверної частини.

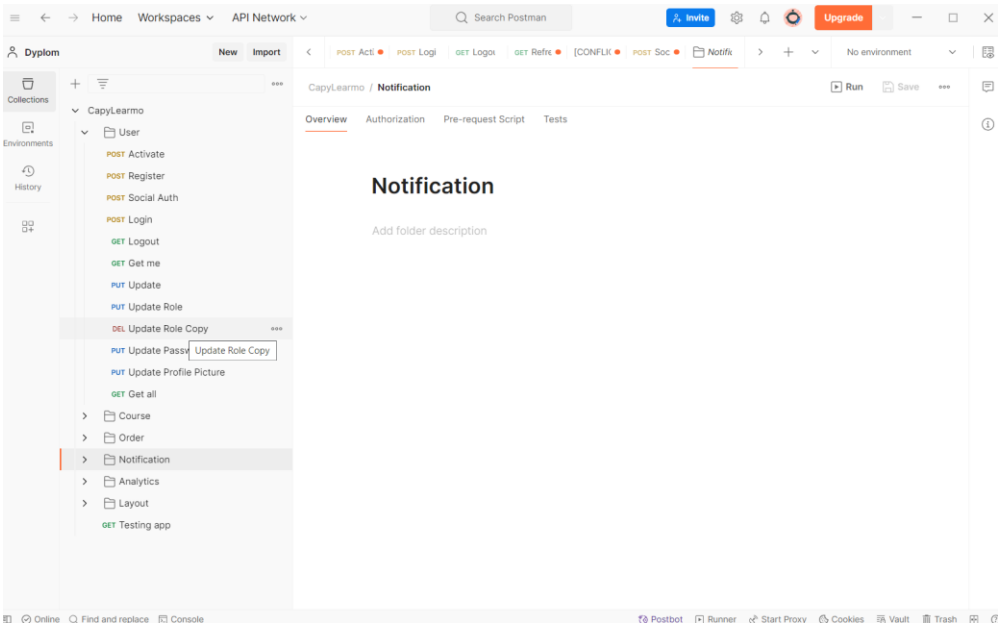


Рисунок 3.1 – Інтерфейс Postman з колекцією CapyLearmo

Папка "User" містить запити, пов'язані з управлінням користувачами в системі CapyLearmo. Основні запити в цій папці включають:

– activate: POST-запит для активації облікового запису користувача;

– register: POST-запит для реєстрації нового користувача;

– login: POST-запит для входу користувача в систему;

– logout: GET-запит для виходу користувача з системи;

– get me: GET-запит для отримання інформації про поточного користувача системи;

– update: PUT-запит для оновлення інформації про користувача;

- update Password: PUT-запит для зміни пароля користувача;
- update Profile Picture: PUT-запит для оновлення аватару користувача.

Папка "Course" містить запити для роботи з курсами в системі CapyLearmo. До цих запитів належать операції створення, оновлення, видалення та отримання інформації про курси.

Папка "Layout" відповідає за управління макетами інтерфейсу користувача. Запити цієї папки дозволяють створювати, змінювати та видаляти макети, а також отримувати інформацію про них.

Папка "Analytics" містить запити для отримання аналітичних даних про використання системи. Запити цієї папки дозволяють отримувати статистику і аналітичні звіти, які допомагають зрозуміти, як користувачі взаємодіють з веб-сайтом для поліпшення користувацького досвіду.

3.1.7 Розміщення серверної частини в мережу Інтернет

У сучасній розробці веб-додатків важливо мати зручний спосіб розгортання серверної частини, який дозволяє швидко й ефективно доставляти оновлення користувачам. Одним із популярних способів досягнення цього є використання платформи Heroku, яка забезпечує простоту розгортання та масштабування додатків. У цьому підрозділі буде описано процес розгортання серверної частини Node.js, написаної на TypeScript, на платформі Heroku [11].

Git [12] — це розподілена система керування версіями, яка дозволяє розробникам відслідковувати зміни у коді та координувати роботу в команді. Спочатку був створений локальний репозиторій Git для управління версіями коду. Локальний репозиторій дозволяє зберігати всі версії проекту, робити коміти та створювати гілки для організації роботи.

Оскільки Heroku не підтримує безпосереднє виконання TypeScript, необхідно скомпілювати код TypeScript у JavaScript [13]. Для цього використовується

TypeScript компілятор. Компіляція перетворює всі файли з розширенням .ts у відповідні файли з розширенням .js, які можуть бути виконані на платформі Heroku.

Після компіляції TypeScript у JavaScript, скомпільовані файли додаються до локального репозиторію Git. Це дозволяє зберігати їх у системі керування версіями та підготувати для завантаження на Heroku.

Heroku — це платформа як послуга, яка дозволяє розробникам швидко розгортати, керувати та масштабувати додатки. Для завантаження проекту на Heroku використовується Heroku CLI. Спочатку створюється новий додаток на Heroku, після чого проект завантажується на платформу.

Heroku автоматично виявляє файл package.json, який містить інформацію про залежності та конфігурацію проекту, та встановлює всі необхідні залежності. Потім він запускає додаток, використовуючи конфігурацію, визначену в цьому файлі.

Після успішного завантаження та розгортання проекту на Heroku, платформа надає посилання на сайт, де можна побачити працюючий додаток. Це посилання можна використовувати для доступу до серверної частини з клієнтської частини.

Розгортання серверної частини Node.js на платформі Heroku є ефективним і зручним способом забезпечення доступності веб додатка для користувачів. Heroku надає широкі можливості автоматизації та масштабування, що суттєво спрощує процес розгортання.

3.2 Розробка клієнстької частини

У цьому підрозділі описано процес розробки клієнтської частини програмного забезпечення, яка відповідає за інтерфейс користувача та взаємодію з серверною частиною. Надається детальний опис реалізації ключових компонентів клієнтської частини.

3.2.1 Сторінки проекту

Головна сторінка (рис. 3.2) відображає загальну інформацію про додаток CapyLearmo, включаючи останні курси, пошуковий рядок, популярні запитання та додаткову інформацію про проект. Вона реалізована у файлі проекту app/page.tsx.

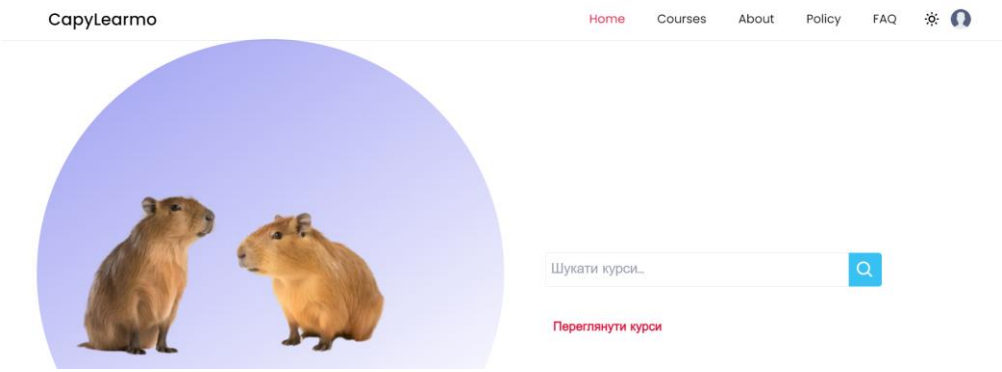


Рисунок 3.2 – Головна сторінка веб-додатку CapyLearmo

На верхній частині сторінки розташований заголовок з логотипом "CapyLearmo" та навігаційне меню, яке включає посилання на такі розділи: Home, Courses, About, Policy, FAQ та іконку користувача для входу в обліковий запис. Центральне місце на сторінці займає велике зображення двох капібар, яке привертає увагу користувачів та створює привабливий візуальний акцент.

Праворуч від головного зображення знаходиться пошуковий рядок, що дозволяє користувачам швидко знайти курси за ключовими словами. Поруч з пошуковим рядком розміщена кнопка "Переглянути курси", яка веде на сторінку з усіма доступними курсами.

Нижче представлено розділ "Популярні запитання", де користувачі можуть знайти відповіді на найпоширеніші. У нижній частині сторінки розташований футер, який містить дві колонки: "Про проект" і "Швидкі посилання". В колонці "Про проект" знаходяться посилання на Політику конфіденційності та інші розділи,

а в колонці "Швидкі посилання" розміщені посилання на курси, обліковий запис користувача та інформаційну панель.

Сторінка профілю (рис. 3.3) дозволяє користувачам переглядати та редагувати свою особисту інформацію, а також взаємодіяти з основними функціями облікового запису. Вона реалізована у файлі app/profile/page.tsx.

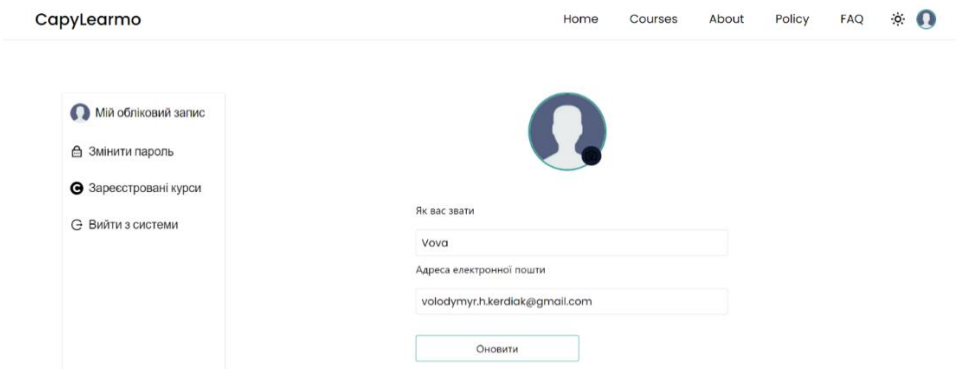


Рисунок 3.3 – Сторінка профілю користувача

Ліворуч на сторінці знаходиться панель навігації облікового запису, яка містить наступні опції:

- відображення загальної інформації про користувача;
- доступ до функції зміни паролю;
- функція для виходу з облікового запису.

Праворуч знаходиться форма для редагування особистої інформації користувача. Тут користувач може змінити своє ім'я та адресу електронної пошти. Над формою розміщено аватар користувача з можливістю завантаження нового зображення. Після внесення змін користувач може натиснути кнопку "Оновити", щоб зберегти зміни.

Сторінка профілю забезпечує користувачам зручний інтерфейс для управління своїм обліковим записом, дозволяючи легко оновлювати особисту інформацію та отримувати доступ до основних функцій системи.

Сторінка курсу (рис. 3.4) забезпечує користувачам детальну інформацію про конкретний курс, включаючи його вміст, опис, відгуки та вартість. Вона реалізована у файлі app/admin/edit-course/[id]/page.tsx.

Під навігаційним меню розташовані категорії курсів, представлені у вигляді кнопок для швидкого фільтрування.

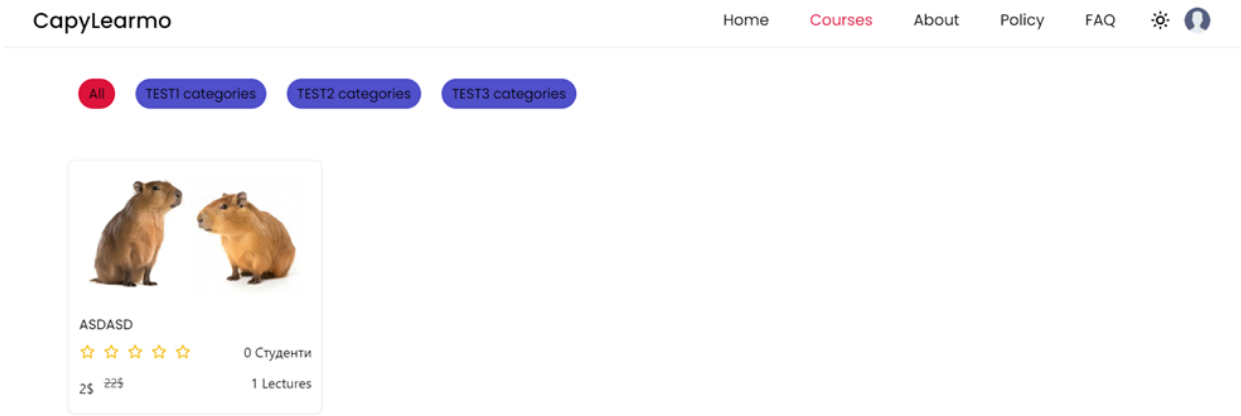


Рисунок 3.4 – Сторінка курсів

На основній частині сторінки розміщений блок з детальною інформацією про курс. У верхній частині блоку знаходиться назва курсу, рейтинг у вигляді зірочок, кількість відгуків та студентів, а також вартість курсу з можливими знижками. Поруч із цією інформацією знаходиться відеоплеєр, який демонструє трейлер або вступне відео курсу. Ця візуальна композиція створює зручний та привабливий інтерфейс для користувачів, дозволяючи їм швидко ознайомитися з ключовою інформацією та отримати уявлення про зміст курсу перед його придбанням.

Під основною інформацією знаходяться деталі курсу, включаючи опис, перелік розділів та уроків, а також відповіді на часті запитання. Наприклад, курс "ASDASD" включає інформацію про те, що ви дізнаєтеся з цього курсу, які передумови потрібні для початку, і детальний огляд змісту курсу (рис. 3.5). Завдяки цьому користувачі можуть отримати повне уявлення про зміст курсу та його відповідність своїм навчальним потребам. Крім того, в розділі з детальною

інформацією розміщене промо-відео, яке надає візуальне уявлення про те, чого можна очікувати від курсу.

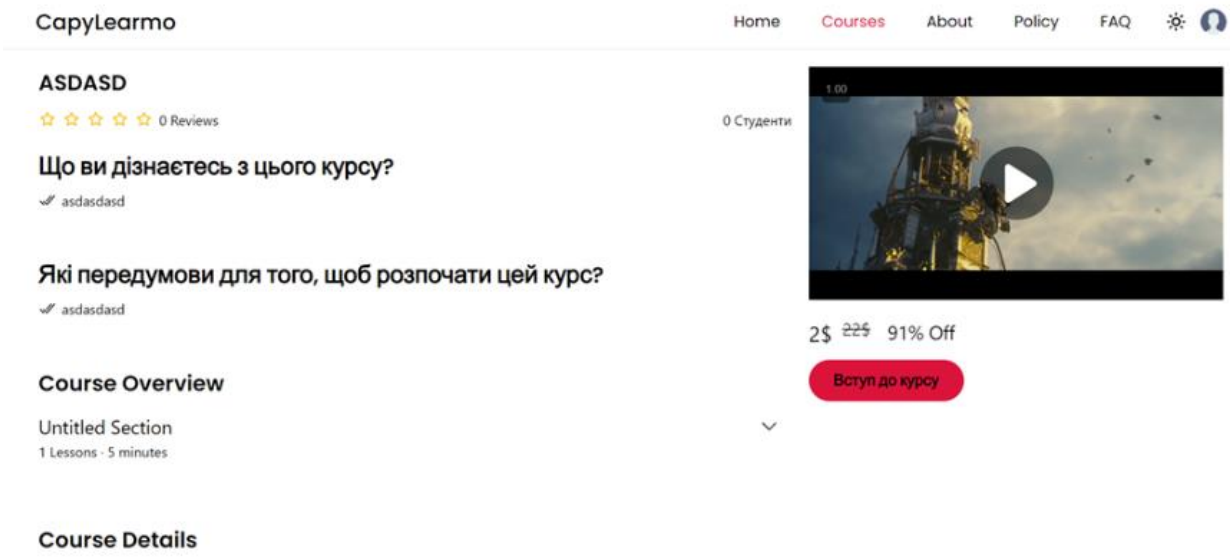


Рисунок 3.5 – Сторінка деталей курсу

Панель адміністратора забезпечує адміністраторам доступ до різних функцій управління додатком, включаючи управління користувачами, курсами, аналітикою та іншими адміністративними завданнями. Вона реалізована у файлі app/admin/page.tsx.

Панель адміністратора складається з декількох ключових розділів, розташованих у боковій навігаційній панелі зліва. Кожен розділ надає доступ до певних функцій та даних доступних адміністратору:

1. Головна панель (рис. 3.6) відображає основну аналітику додатка, включаючи графіки користувацької активності, кількість нових користувачів та отримані продажі. На цій сторінці також показуються останні транзакції, що дозволяє адміністраторам швидко переглянути останні дії користувачів.

Рисунок 3.6 – Головна панель адміністратора

2. Користувачі (рис. 3.7) дозволяє адміністраторам переглядати та керувати користувачами системи. Тут можна бачити список користувачів, їхні ролі та дату приєднання. Адміністратори можуть давати ролі користувачам, змінювати їхні існуючі ролі, а також переглядати додаткову інформацію, таку як статус активності та останню дату входу в систему. Це забезпечує ефективне управління користувачами, дозволяючи адміністраторам швидко приймати рішення щодо доступу та прав користувачів у системі.

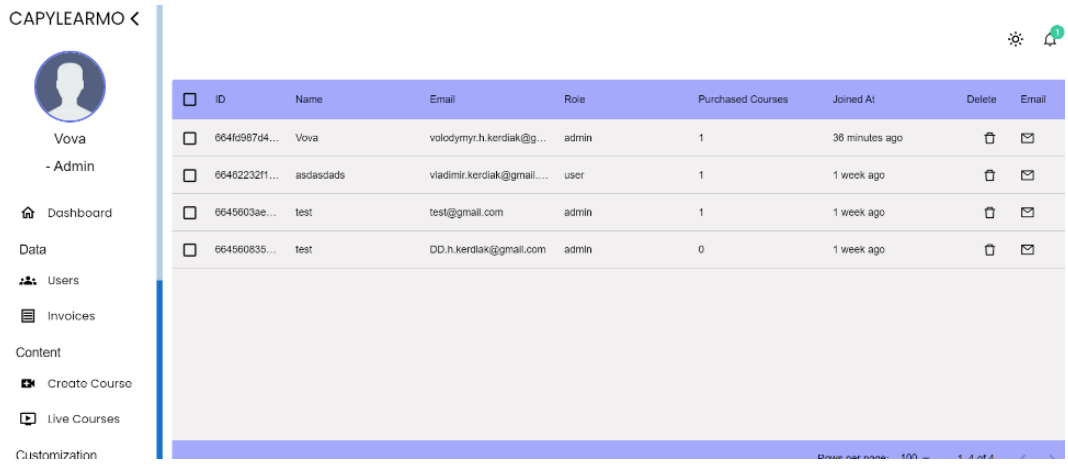


Рисунок 3.7 – Розділ користувачі

Створення курсу (рис. 3.8) надає інтерфейс для створення та редагування курсів. Адміністратори можуть вказати назву курсу, опис, ціну, теги, категорії та

завантажити зображення. Після заповнення необхідної інформації курс можна опублікувати, щоб зробити його доступним для користувачів системи. Інтерфейс також включає можливість додавання розділів та уроків до курсу, що дозволяє структурувати навчальний матеріал у логічній послідовності. Інтерфейс для створення курсу є зрозумілим та зручним, що спрощує процес додавання нових курсів та управління вже існуючими. Цей інтерфейс забезпечує гнучкість у налаштуванні курсів та якість подання матеріалу.

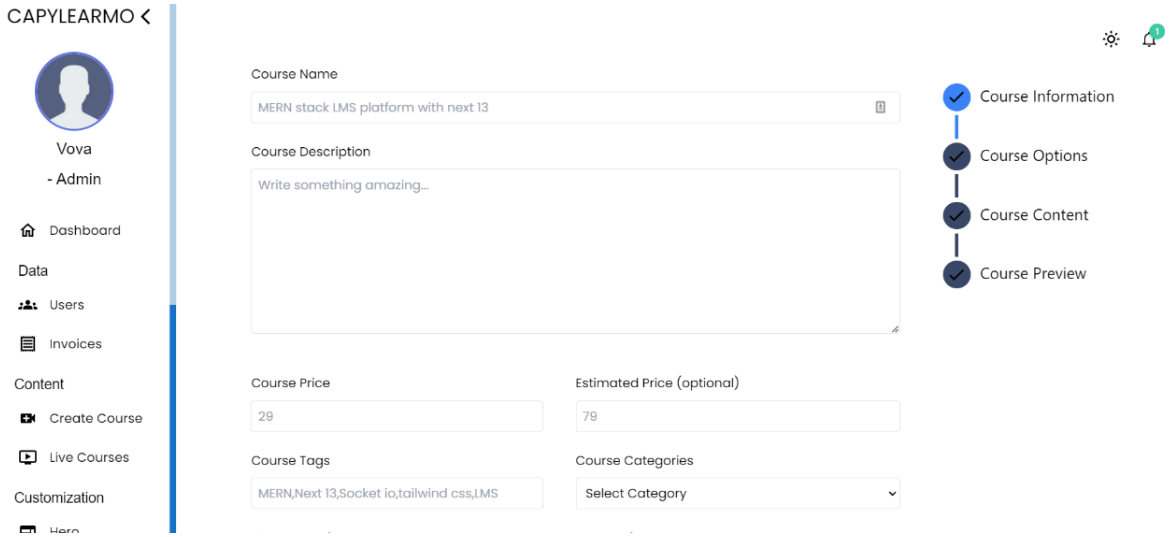


Рисунок 3.8 — Процес створення курсу

3. Курси (рис. 3.9) дозволяє адміністраторам переглядати список усіх курсів, редагувати їх або видаляти. Тут можна бачити назву курсу, його рейтинг, кількість придбань, а також дату створення, що забезпечує зручний та ефективний процес управління курсами та аналізу їхньої активності. Адміністратори можуть швидко втручатися та вносити зміни в курси, відповідно до потреб та вимог.

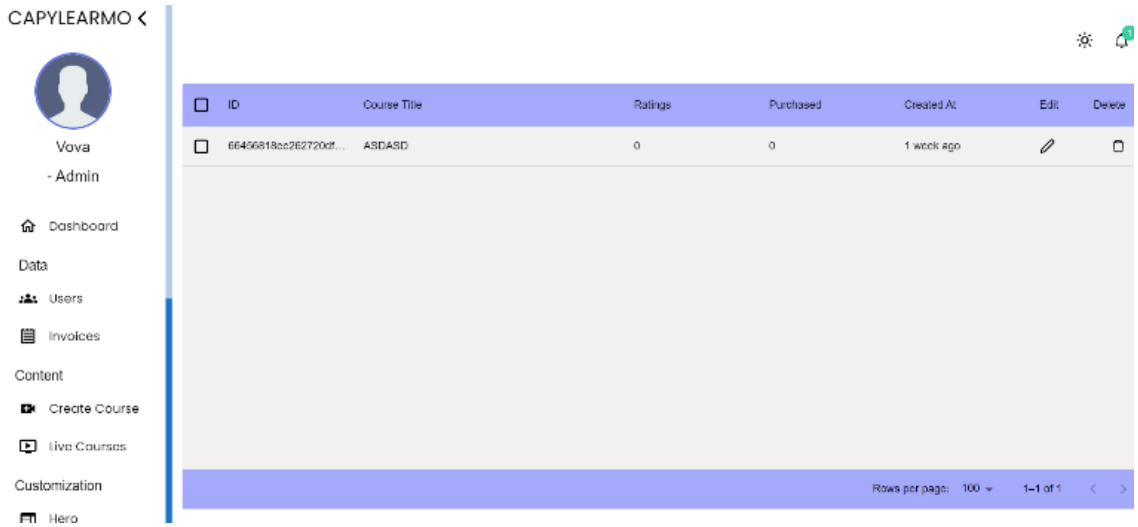


Рисунок 3.9 – Розділ з інформацією про курси

4. Команда (рис. 3.10) дозволяє адміністраторам керувати командою, додаючи нових членів або редагуючи інформацію про існуючих. Тут відображається список членів команди, їхні ролі та контактна інформація, що забезпечує зручний та ефективний процес управління персоналом. Адміністратори можуть легко вносити зміни в команду, а також відслідковувати її склад та зміни в ролях та обов'язках кожного члена.

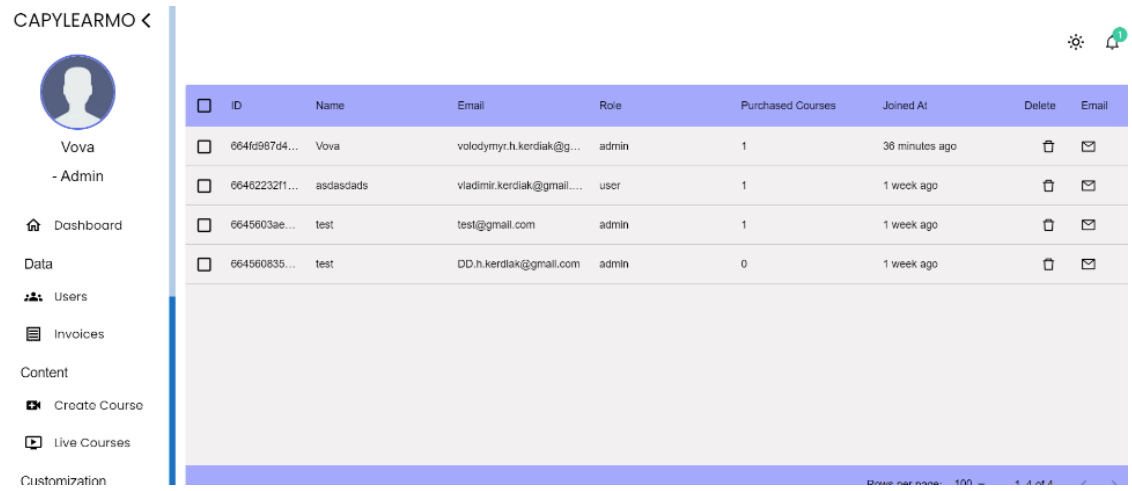


Рисунок 3.10 – Розділ команда

5. Аналітика містить кілька підрозділів для аналізу даних про курси та користувачів. Це включає графіки та звіти про активність користувачів, курсів та інші важливі показники.

6. Налаштування дозволяє адміністраторам налаштовувати вигляд додатка, включаючи редагування часто задаваних питань (FAQ) та налаштування категорій курсів.

Панель адміністратора забезпечує зручний та інтуїтивно зрозумілий інтерфейс для ефективного управління додатком, дозволяючи адміністраторам легко виконувати всі необхідні завдання та отримувати доступ до інформації.

3.2.2 Розміщення для загального доступу

Для розгортання клієнтської частини використовується платформа Vercel, яка забезпечує швидке та зручне розгортання Next.js додатків. Vercel автоматично виявляє конфігурацію проекту, виконує необхідні кроки для його розгортання, а також забезпечує автоматичне масштабування і оптимізацію продуктивності додатку. Крім того, Vercel підтримує безперервну інтеграцію та розгортання (CI/CD), дозволяючи автоматично оновлювати додаток при кожному внесенні змін до репозиторію. Платформа також надає зручний інтерфейс для моніторингу та управління розгорнутими проектами, що спрощує підтримку та подальший розвиток клієнтської частини.

Першим кроком є створення нового проекту на платформі Vercel. Це можна зробити через веб-інтерфейс Vercel, де користувачеві пропонується підключити свій Git репозиторій. Vercel підтримує інтеграцію з різними системами керування версіями, такими як GitHub [17], GitLab [16], та Bitbucket [15]. Після входу в обліковий запис Vercel [14], користувач підключає свій репозиторій, вибираючи його зі списку доступних репозиторіїв (рис 3.11).

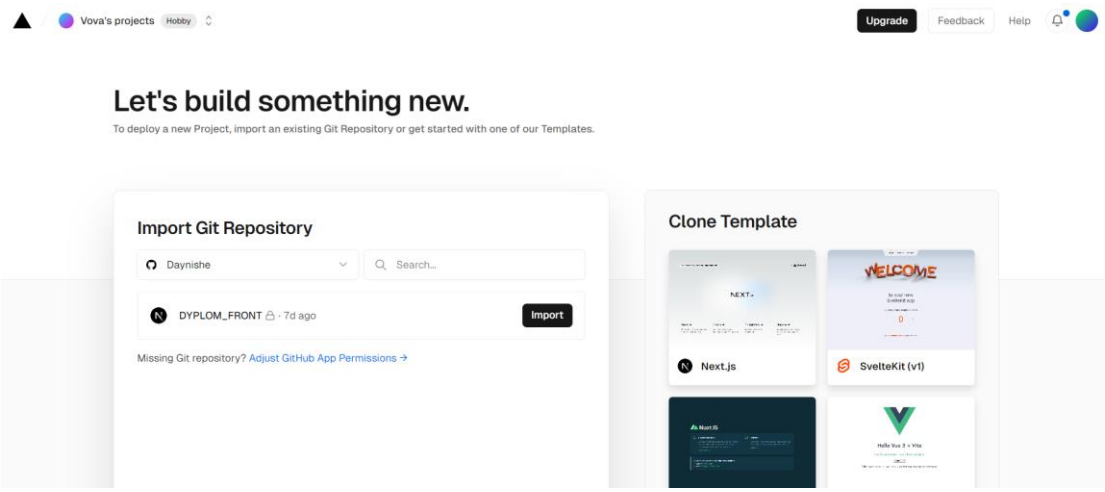


Рисунок 3.11 – веб-інтерфейс для підключення репозиторію

Vercel забезпечує кілька важливих переваг для розгортання клієнтських додатків на їхньому ресурсі:

– простота розгортання: мінімальні налаштування та автоматичне розгортання проекту, що дозволяє швидко доставляти зміни користувачам;

– швидкість: висока швидкість завантаження сторінок завдяки серверному рендерингу та оптимізації ресурсів;

– інтеграції: підтримка інтеграцій з GitHub, GitLab та іншими системами керування версіями, що полегшує процес безперервної інтеграції та розгортання;

– моніторинг та аналітика: Vercel надає інструменти для моніторингу продуктивності додатку та отримання аналітичних даних.

Після створення проекту на Vercel та підключення репозиторію, розпочинається процес налаштування та розгортання. Користувач може вибрати гілку репозиторію, яка буде використовуватися для розгортання. Зазвичай це основна гілка (main або master). Vercel автоматично виявляє файл next.config.js, який містить налаштування Next.js проекту, та запускає процес розгортання. Це включає встановлення залежностей, збірку проекту та розгортання на Vercel.

Після завершення процесу розгортання Vercel надає посилання на працюючий додаток, який можна переглянути у браузері. За цим посилання можна отримати доступ до клієнтської частини, що розміщена на Vercel платформі.

Висновки до розділу 3

У цьому розділі детально описано процес розробки серверної та клієнтської частини програмного забезпечення для управління освітніми курсами, їхніми матеріалами, а також інтеракції з користувачами. Було розглянуто основні аспекти розробки та розміщення веб-застосунку в мережі Інтернет, що дозволило створити ефективний, зручний та безпечний інструмент для навчання.

Розробка серверної частини включала створення та опис схем даних для зберігання інформації про користувачів, курси та макети в базі даних MongoDB з використанням бібліотеки mongoose. Було реалізовано бізнес-логіку додатку у вигляді сервісів, які забезпечують виконання основних операцій, таких як створення курсів, управління користувачами та оновлення ролей. Розроблено контролери для обробки HTTP-запитів, які здійснюють взаємодію між клієнтською частиною та сервісами. Використано утиліти для генерації аналітичних даних, обробки помилок, роботи з JWT, Redis та електронною поштою, а також проміжне програмне забезпечення (middleware) для забезпечення аутентифікації, авторизації та обробки асинхронних помилок. Для тестування серверної частини було використано Postman, що дозволило автоматизувати тестування, створювати документацію та інтегруватися з CI/CD пайплайнами. Процес розгортання серверної частини на платформі Heroku включав налаштування Git, компіляцію TypeScript у JavaScript та автоматичне встановлення залежностей.

Розробка клієнтської частини охоплювала створення ключових сторінок інтерфейсу користувача, таких як головна сторінка, сторінка профілю, сторінка курсу та панель адміністратора. Було забезпечено доступ до функцій управління

додатком, включаючи аналітику, управління користувачами та курсами, а також налаштування додатку. Для розміщення клієнтської частини використовувалася платформа Vercel, яка забезпечила автоматичне налаштування та розгортання проекту з інтеграцією з системами керування версіями та надання інструментів для моніторингу продуктивності.

Цей розділ демонструє важливість інтегрованого підходу до розробки, тестування та розгортання веб-застосунків, що дозволяє створити надійний та зручний продукт для кінцевих користувачів.

ВИСНОВКИ

У рамках цієї роботи було проведено всебічне дослідження в області управління освітніми матеріалами та навчальними курсами. Було розроблено веб-сайт, що спрямований на підвищення ефективності взаємодії між викладачами та студентами, а також оптимізацію процесу управління навчальним контентом.

Огляд існуючих рішень: було виконано аналіз існуючих платформ і засобів для управління освітніми ресурсами, який показав, що багато із них не відповідають сучасним вимогам щодо гнучкості, масштабованості та інтеграційної взаємодії.

Аналіз проблеми: детальний аналіз потреб освітніх установ дозволив визначити основні виклики, з якими стикаються користувачі — від недостатньої інтерактивності до складнощів у керуванні та розподілі навчальних матеріалів.

Постановка задачі: на основі аналізу були сформульовані конкретні завдання для розробки веб-сайту, який би вирішував вищевказані проблеми, забезпечуючи легкість доступу до матеріалів та їх ефективне управління.

Визначення шляхів реалізації: були обрані найбільш підходящі технології та методики розробки, які включають використання Node.js для серверної частини та Next.js для клієнтської частини, що забезпечує високу продуктивність та гнучкість системи при різних умовах використання.

Використання технологій: розробка серверної частини на .NET та клієнстької на React дозволила створити масштабовану архітектуру з зручним, інтуїтивно зрозумілим інтерфейсом.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Перша в Україні дистанційна платформа громадянської освіти - ВУМ online. URL: https://vumonline.ua/ (дата звернення: 28.03.2024).

2. Учасники проектів Вікімедіа. Git – вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Git (дата звернення: 23.05.2024).

3. Учасники проектів Вікімедіа. Node.js – вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Node.js (дата звернення: 28.03.2024).

4. Учасники проектів Вікімедіа. React – вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/React (дата звернення: 23.05.2024).

5. About heroku. Cloud Application Platform | Heroku. URL: https://www.heroku.com/about (дата звернення: 28.03.2024).

6. About us -- Vercel. Vercel. URL: https://vercel.com/about (дата звернення: 28.03.2024).

7. Bitbucket. Bitbucket. URL: https://bitbucket.org/ (дата звернення: 28.03.2024).

8. Build software better, together. GitHub. URL: https://github.com/about (дата звернення: 28.03.2024).

9. Coursera. Coursera. URL: https://www.coursera.org/ (дата звернення: 28.03.2024).

10. Documentation - TypeScript for JavaScript Programmers. TypeScript: JavaScript With Syntax For Types. URL: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html (дата звернення: 28.03.2024).

11. EdEra – студія онлайн-освіти. EdEra. URL: https://ed-era.com/ (дата звернення: 28.03.2024).

12. Git. Git. URL: https://git-scm.com/ (дата звернення: 28.03.2024).

13. JavaScript | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 28.03.2024).

14. MongoDB: the developer data platform. MongoDB. URL: https://www.mongodb.com/ (дата звернення: 28.03.2024).

15. Next.js on vercel. Vercel. URL: https://vercel.com/frameworks/nextjs?utm_source=next_site&utm_medium=showcase_redesign&utm_campaign=hero_cta (дата звернення: 28.05.2024).

16. Node.js – about node.js. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/en/about (дата звернення: 28.03.2024).

17. Prometheus – Найбільша платформа онлайн-курсів в Україні. Prometheus. URL: https://prometheus.org.ua/ (дата звернення: 28.03.2024).

18. Tailwind CSS. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. URL: https://tailwindcss.com/docs/installation (дата звернення: 28.03.2024).

19. The most-comprehensive AI-powered DevSecOps platform. The most-comprehensive AI-powered DevSecOps platform | GitLab. URL: https://about.gitlab.com/ (дата звернення: 28.03.2024).

20. Postman – Найбільша платформа для тестування API. URL: https://www.postman.com/ (дата звернення: 28.03.2024).

ДОДАТКИ

Додаток А

Схема mongoose для моделі курса

```
import mongoose, { Document, Model, Schema } from "mongoose";
import { IUser } from "./user.model";

export interface IComment extends Document {
  user: IUser;
  question: string;
  questionReplies: IComment[];
}

interface IReview extends Document {
  user: IUser;
  rating?: number;
  comment: string;
  commentReplies?: IReview[];
}

interface ILink extends Document {
  title: string;
  url: string;
}

interface ICourseData extends Document {
  title: string;
  description: string;
  videoUrl: string;
  videoThumbnail: object;
  videoSection: string;
  videoLength: number;
```

```
  videoPlayer: string;
  links: ILink[];
  suggestion: string;
  questions: IComment[];
}

 export interface ICourse extends Document {
  name: string;
  description: string;
  categories: string;
  price: number;
  estimatedPrice?: number;
  thumbnail: object;
  tags: string;
  level: string;
  demoUrl: string;
  benefits: { title: string }[];
  prerequisites: { title: string }[];
  reviews: IReview[];
  courseData: ICourseData[];
  ratings?: number;
  purchased: number;
}

const reviewSchema = new Schema<IReview>({
  user: Object,
  rating: {
    type: Number,
    default: 0,
  },
  comment: String,
  commentReplies: [Object],
},{timestamps:true});

const linkSchema = new Schema<ILink>({
  title: String,
```

```
  url: String,
});

const commentSchema = new Schema<IComment>({
  user: Object,
  question: String,
  questionReplies: [Object],
},{timestamps:true});

const courseDataSchema = new Schema<ICourseData>({
  videoUrl: String,
  videoThumbnail: Object,
  title: String,
  videoSection: String,
  description: String,
  videoLength: Number,
  videoPlayer: String,
  links: [linkSchema],
  suggestion: String,
  questions: [commentSchema],
});

const courseSchema = new Schema<ICourse>({
  name: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  categories:{
    type:String,
    required: true,
  },
  price: {
```

```
  type: Number,
  required: true,
},
estimatedPrice: {
  type: Number,
},
thumbnail: {
  public_id: {
    type: String,
  },
  url: {
    type: String,
  },
},
tags:{
  type: String,
  required: true,
},
level:{
  type: String,
  required: true,
},
demoUrl:{
  type: String,
  required: true,
},
benefits: [{title: String}],
prerequisites: [{title: String}],
reviews: [reviewSchema],
 courseData: [courseDataSchema],
 ratings:{
   type: Number,
   default: 0,
 },
 purchased:{
  type: Number,
```

```
    default: 0,
   },
},{timestamps: true});


const CourseModel: Model<ICourse> = mongoose.model("Course", courseSchema);

export default CourseModel;
```

Додаток Б

Сервіс для роботи з курсами

```
import { Response } from "express";
import CourseModel from "../models/course.model";
import { CatchAsyncError } from "../middleware/catchAsyncErrors";
// create course
export const createCourse = CatchAsyncError(async(data:any,res:Response)=>{
const course = await CourseModel.create(data);
res.status(201).json({
success:true,
course
});
})
// Get All Courses
export const getAllCoursesService = async (res: Response) => {
const courses = await CourseModel.find().sort({ createdAt: -1 });
res.status(201).json({
success: true,
courses,
});
};
```

```
 import { Response } from "express";
import { redis } from "../utils/redis";
import userModel from "../models/user.model";
```

```
export const getUserById = async (id: string, res: Response) => {
  const userJson = await redis.get(id);

  if (userJson) {
    const user = JSON.parse(userJson);
    res.status(201).json({
      success: true,
      user,
    });
  }
};


export const getAllUsersService = async (res: Response) => {
  const users = await userModel.find().sort({ createdAt: -1 });

  res.status(201).json({
    success: true,
    users,
  });
};

export const updateUserRoleService = async (res:Response,id:
string,role:string) => {
  const user = await userModel.findByIdAndUpdate(id, { role }, { new: true
});

  res.status(201).json({
    success: true,
    user,
  });
}
```

метадані

Заголовок

Розробка веб-сайту для управління освітніми матеріалами та курсами навчання

Автор
Кердяк В. Г.

Науковий керівник / Експерт
кандидат технічних наук Олег Пашкевич

підрозділ
King Danylo University

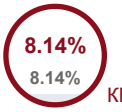
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

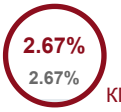
Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	1
Білі знаки	ß	0
Парафрази (SmartMarks)	a	72

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

9676
Кількість слів

77371
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/396/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D0%A1%D1%82%D1%80%D1%96%D0%BB%D0%B5%D1%86%D1%8C%D0%BA%D0%B8%D0%B9.pdf?sequence=1	52	0.54 %
2	http://repository.ukd.edu.ua/bitstream/handle/123456789/396/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D0%A1%D1%82%D1%80%D1%96%D0%BB%D0%B5%D1%86%D1%8C%D0%BA%D0%B8%D0%B9.pdf?sequence=1	40	0.41 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/390/%D0%9C%D0%B0%D0%BD%D1%82%D1%83%D0%BB%D1%8F%D0%BA%20%D0%94.%D0%92.%20%D0%9A%D0%A0.pdf?sequence=1	38	0.39 %