Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
бакалавра

на тему: «ДОСЛІДЖЕННЯ ІНСТРУМЕНТІВ ФРЕЙМВОРКІВ VUE ТА LARAVEL ДЛЯ ПРОЄКТУВАННЯ ТА РОЗРОБКИ САЙТУ»

Виконав: студент 3 курсу, KNms1-B21 групи
спеціальності 122 “Комп’ютерні науки”
Ігнатов Максим Васильович
Керівник:
Моцик Ростислав Васильович
доцент кафедри комп’ютерних наук,
кандидат педагогічних наук, доцент
Рецензент:
Сморжевський Ю.Л.
доцент кафедри математики,
кандидат педагогічних наук, доцент

Кам’янець-Подільський 2024 р.

АНОТАЦІЯ

Ця дипломна робота присвячена дослідженню можливостей фреймворків Vue.js та Laravel для проектування і розробки веб-застосунку. У рамках роботи було проведено аналіз архітектури та функціональних можливостей обох фреймворків, розглянуто їх інтеграцію та взаємодію для створення сучасного та ефективного веб-застосунку. Основною метою дипломної роботи є розробка онлайн платформи для навчання, яка забезпечує зручний інтерфейс для користувачів, можливість перегляду та реєстрації на курси, управління контентом для адміністраторів, а також безпечне зберігання і обробку даних. У ході роботи були розглянуті питання побудови користувацького інтерфейсу за допомогою Vue.js, організації бізнес-логіки та роботи з базою даних з використанням Laravel. Проект демонструє переваги використання даних технологій для створення веб-додатків з високою продуктивністю, масштабованістю та зручністю у підтримці. Окрема увага приділена питанням безпеки, оптимізації роботи додатку та тестуванню його функціональності. Результати даного дослідження можуть бути корисними для розробників, які бажають використовувати Vue.js та Laravel у своїх проектах, а також для студентів та викладачів, які цікавляться сучасними підходами до веб-розробки.

ABSTRACT

This thesis is dedicated to exploring the capabilities of the Vue.js and Laravel frameworks for designing and developing a web application. The work involves an analysis of the architecture and functional capabilities of both frameworks, examining their integration and interaction to create a modern and efficient web application. The primary objective of the thesis is to develop an online learning platform that provides a user-friendly interface for users, the ability to view and register for courses, content management for administrators, and secure data storage and processing. The work addresses the construction of the user interface using Vue.js, the organization of business logic, and database management using Laravel. The project demonstrates the advantages of using these technologies to create web applications with high performance, scalability, and ease of maintenance. Special attention is given to security issues, application performance optimization, and functional testing. The results of this study can be valuable for developers looking to use Vue.js and Laravel in their projects, as well as for students and educators interested in modern web development approaches.

ЗМІСТ

ВСТУП

З кожним днем веб-технології розвиваються, з'являються нові вимоги та можливості. У зв'язку з цим, розробники веб-сайтів мають постійно оновлювати свої навички та знання, а також використовувати сучасні інструменти та технології. Vue.js та Laravel є двома із найбільш високопродуктивних інструментів у своїх областях, які пропонують ефективні рішення для веб-розробки. Дослідження їхніх можливостей та вивчення їхніх переваг та недоліків стає дуже актуальним завданням для будь-якого веб-розробника.

Актуальність теми: полягає у зростаючій потребі у високоякісних онлайн платформах для навчання, що можуть забезпечити ефективну взаємодію користувачів та інструкторів. Сучасні освітні технології вимагають від платформ надійності, безпеки та продуктивності, що робить використання таких фреймворків як Vue.js і Laravel важливим для досягнення цих цілей. Дослідження та розробка ефективної платформи сприятиме підвищенню якості онлайн навчання та задоволенню потреб сучасних користувачів.

Мета полягає у дослідженні інструментів фреймворків Vue.js та Laravel для проєктування та розробки сучасного веб-сайту.

Завдання роботи:

- дослідити та проаналізувати вимоги користувачів і бізнес-вимог до платформи, визначити ключові функції для успішного функціонування.

- спроектувати архітектуру системи, включаючи структуру бази даних, моделі, контролери та маршрути, а також розробка API для взаємодії між фронтендом і бекендом.

- створити зручний та функціональний інтерфейс користувача, включаючи реєстрацію, авторизацію, перегляд та пошук курсів, управління профілем.

- впровадження механізмів для забезпечення безпеки даних користувачів, запобігання SQL-ін'єкціям, забезпечення безпеки авторизації,

оптимізація продуктивності додатку.

- провести юніт-тестування, інтеграційного та функціонального тестування додатку, виправлення знайдених помилок та оптимізація коду.

Об'єкт дослідження: інтеграція фреймворків Vue.js та Laravel для розробки сучасних веб-додатків.

Предмет дослідження: функціональні можливості фреймворків Vue.js і Laravel при розробці платформи для курсів та навчання онлайн.

Новизна теми полягає у комплексному дослідженні та аналізі сучасних фреймворків Vue.js та Laravel для проєктування та розробки веб-сайтів, що поєднує як теоретичні, так і практичні аспекти їхнього застосування. Хоча кожен з цих фреймворків окремо добре вивчений і широко використовується в галузі веб-розробки, їх інтеграція для створення єдиного, високоефективного та динамічного веб-додатку є відносно новим підходом, який потребує додаткового аналізу та оцінки. Дослідження нових можливостей та викликів, що виникають при поєднанні Vue.js для фронтенд-розробки та Laravel для бекенд-розробки, має на меті розкрити переваги такого підходу та запропонувати ефективні рішення для оптимізації процесу розробки. Робота також включає аналіз сучасних тенденцій у веб-розробці та дослідження впливу цих фреймворків на продуктивність, масштабованість та зручність веб-додатків.

Структура роботи. Дипломна робота складається зі вступу, трьох розділів, висновку, списку використаних джерел.

1. ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА МОЖЛИВОСТІ ФРЕЙМВОРКІВ

1.1 Огляд сучасних технологій для розробки веб-додатків, функціональні можливості та обмеження

Інтернет технології пройшли значний шлях від своїх витоків до сучасного стану, де веб-додатки відіграють ключову роль у різних сферах діяльності. Розробка веб-додатків включає дві основні складові: фронтенд (клієнтська частина) та бекенд (серверна частина).

Фронтенд-технології забезпечують взаємодію користувача з додатком. HTML (HyperText Markup Language) надає структуру веб-сторінок, CSS (Cascading Style Sheets) відповідає за їх стильове оформлення, а JavaScript додає інтерактивність і динаміку. Сучасні фронтенд-фреймворки, такі як React, Angular та VUE, значно спрощують розробку складних користувацьких інтерфейсів, забезпечуючи компонентний підхід та управління станом додатка.

Бекенд-технології забезпечують обробку даних, виконання бізнес-логіки та взаємодію з базами даних. PHP, Python, Node.js та Ruby є популярними мовами програмування для серверної частини, кожна з яких має свої фреймворки: LARAVEL для PHP, Django для Python, Express для Node.js та Rails для Ruby. Ці фреймворки допомагають розробникам швидко створювати надійні та масштабовані додатки.

Сучасні тенденції у веб-розробці включають односторінкові додатки (SPA), прогресивні веб-додатки (PWA) та JAMstack архітектуру. SPA забезпечують високу швидкість та зручність використання завдяки динамічному завантаженню контенту без перезавантаження сторінки. PWA поєднують переваги веб- та мобільних додатків, надаючи можливість роботи офлайн та підтримку push-сповіщень. JAMstack (JavaScript, APIs, Markup) архітектура сприяє підвищенню продуктивності та безпеки завдяки відокремленню клієнтської та серверної частин.

Vue.js та Laravel є потужними інструментами для розробки сучасних веб-додатків, кожен з яких має свої унікальні функціональні можливості та обмеження.

Vue.js – це прогресивний JavaScript фреймворк для побудови користувацьких інтерфейсів. Він надає високу гнучкість та дозволяє поступово впроваджувати свої компоненти в існуючий код. Серед основних функціональних можливостей Vue.js можна виділити реактивність, що забезпечує автоматичне оновлення інтерфейсу при зміні стану даних, та компонентну архітектуру, яка сприяє модульності та повторному використанню коду[2]. Vue.js також відомий своєю легкою вагою та швидкістю, що дозволяє створювати високопродуктивні додатки. До того ж, він має відмінну документацію та активну спільноту розробників, що полегшує процес навчання та вирішення проблем[3].

Однак, Vue.js має деякі обмеження. Наприклад, він не включає в себе стандартних рішень для управління станом або маршрутизації, тому для побудови більш складних додатків потрібні додаткові бібліотеки, такі як Vuex для управління станом та Vue Router для маршрутизації. Крім того, через свою новизну, Vue.js може не мати такого широкого екосистемного оточення та кількості готових рішень, як інші більш зрілі фреймворки.

Laravel – це PHP фреймворк, що спеціалізується на розробці бекенд частини додатків. Він надає потужні інструменти для реалізації задач, пов'язаних з обробкою запитів, управлінням базами даних, аутентифікацією користувачів та багато іншого. Laravel підтримує модульну архітектуру та використання готових пакетів, що значно спрощує процес розробки. Він також включає такі інструменти як Eloquent ORM для роботи з базами даних, Blade шаблонізатор для генерації HTML, та потужну систему маршрутизації[1].

Серед обмежень Laravel варто зазначити його відносно високу складність для початківців, що може ускладнити початковий етап навчання. Крім того, Laravel є монолітним фреймворком, що може бути обмеженням для розробників, які надають перевагу мікросервісній архітектурі. Також, як і у

випадку з Vue.js, Laravel потребує додаткових налаштувань та конфігурацій для інтеграції з фронтенд фреймворками та іншими сторонніми бібліотеками.

Таким чином, Vue.js та Laravel забезпечують потужні функціональні можливості для розробки веб-додатків, доповнюючи один одного. Однак, розробники повинні враховувати їхні обмеження при виборі технологій для своїх проєктів.

1.2 Порівняння з іншими фреймворками та технологічні основи

Порівняння Vue.js та Laravel з іншими фреймворками надає глибше розуміння їхніх сильних сторін та обмежень, що дозволяє зробити усвідомлений вибір при розробці веб-додатків.

Vue.js, React та Angular є одними з найбільш популярних JavaScript фреймворків для фронтенд-розробки. Vue.js виділяється своєю простотою та гнучкістю, дозволяючи поступово інтегрувати його в існуючі проєкти. Він пропонує реактивність та компонентну архітектуру, що спрощує розробку динамічних інтерфейсів. React, розроблений компанією Facebook, також фокусується на компонентному підході та має велику екосистему, що включає такі інструменти як React Router для маршрутизації та Redux для управління станом. React є більш популярним у великих корпоративних проєктах завдяки своїй потужній підтримці та стабільності. Angular, розроблений Google, є комплексним фреймворком, що надає всі необхідні інструменти з коробки, включаючи вбудований модуль для управління станом, маршрутизацію та форми. Angular є важчим у навчанні через свою складність, але він забезпечує високу продуктивність та масштабованість[7].

Laravel, Ruby on Rails та Django є популярними бекенд-фреймворками. Laravel, написаний на PHP, відомий своєю елегантністю та простотою використання. Він має потужний ORM (Eloquent), зручну систему маршрутизації та шаблонізатор Blade. Ruby on Rails, написаний на Ruby, має схожий підхід до розробки з фокусом на швидкість і простоту написання коду.

Rails також використовує ORM (ActiveRecord) та забезпечує високий рівень автоматизації рутинних задач. Django, фреймворк для Python, відомий своєю надійністю та масштабованістю. Django забезпечує безпечність та простоту розробки через використання вбудованих адміністративних інтерфейсів та системи ORM.

Vue.js у порівнянні з React та Angular є менш складним у навчанні, що робить його привабливим для новачків та малих проєктів. Він дозволяє швидко створювати прототипи та інтегруватися з іншими технологіями. Однак, React має більшу екосистему та більш стабільну підтримку для великих корпоративних проєктів, тоді як Angular надає повний стек інструментів, але потребує більше зусиль для освоєння.

Laravel, у порівнянні з Ruby on Rails та Django, пропонує сучасні можливості та гарну документацію, що робить його популярним серед PHP розробників. Він надає більшу свободу у виборі структур та підходів у порівнянні з Rails та Django, які мають більш строгі конвенції. Django є відмінним вибором для проєктів, що потребують високої безпеки та масштабованості, а Ruby on Rails — для швидкої розробки MVP (мінімально життєздатного продукту)[8].

Таким чином, вибір між Vue.js, React, Angular, Laravel, Ruby on Rails та Django залежить від конкретних вимог проєкту, наявних ресурсів та навичок команди розробників. Кожен з цих фреймворків має свої сильні сторони, які можуть бути ключовими у визначенні успішності проєкту.

Технологічні основи розробки веб-додатків з використанням Vue.js та Laravel включають широкий спектр інструментів та методологій, які забезпечують ефективність, масштабованість та гнучкість розробки.

Vue.js є прогресивним JavaScript фреймворком, який фокусується на побудові користувацьких інтерфейсів. Його основою є реактивна система, яка дозволяє автоматично оновлювати інтерфейс у відповідь на зміни стану даних. Це досягається за допомогою об'єктів `Vue` та методів `watch`, які відстежують зміни у властивостях і автоматично рендерять відповідні компоненти.

Компонентна архітектура Vue.js дозволяє розбивати застосунок на невеликі, незалежні частини, які можна повторно використовувати та комбінувати. Для керування станом додатку використовується бібліотека Vuex, яка забезпечує централізоване сховище даних та полегшує управління складними станами. Для маршрутизації у Vue.js застосовується Vue Router, що дозволяє створювати динамічні маршрути та обробляти навігацію в додатку.

Laravel є потужним PHP фреймворком для розробки серверних додатків. Основою Laravel є його елегантна та проста у використанні синтаксична конструкція, яка робить код читабельним та зрозумілим. Один з ключових компонентів Laravel — це Eloquent ORM, який забезпечує зручну роботу з базами даних через об'єктно-реляційну модель. Він дозволяє виконувати складні запити та маніпулювати даними, використовуючи простий та інтуїтивно зрозумілий синтаксис. Laravel також має потужну систему маршрутизації, яка дозволяє легко визначати маршрути для HTTP-запитів та обробляти їх. Шаблонізатор Blade, що входить до складу Laravel, забезпечує гнучкий та зручний спосіб генерації HTML-коду з використанням шаблонів.

Інтеграція Vue.js з Laravel забезпечує створення потужних та динамічних веб-додатків. Laravel може використовувати Vue.js для побудови сучасних користувацьких інтерфейсів, що дозволяє розробникам отримати переваги обох технологій. Наприклад, Vue.js може використовуватися для створення динамічних компонентів інтерфейсу, які взаємодіють з API, побудованим за допомогою Laravel. Це забезпечує розподіл логіки між клієнтом та сервером, що покращує продуктивність та масштабованість додатків[4].

Розробка з використанням Vue.js та Laravel також включає застосування сучасних інструментів та технологій, таких як Webpack для зборки модулів, Docker для контейнеризації додатків та CI/CD (безперервна інтеграція та доставка) для автоматизації процесу розгортання. Webpack дозволяє оптимізувати та мінімізувати ресурси, забезпечуючи швидке завантаження додатків. Docker забезпечує ізоляцію середовища розробки та полегшує розгортання додатків у різних середовищах. CI/CD підходи дозволяють

автоматизувати процеси тестування та розгортання, забезпечуючи високу якість та стабільність програмного забезпечення.

Таким чином, технологічні основи розробки з використанням Vue.js та Laravel включають сучасні методології та інструменти, що забезпечують ефективну та масштабовану розробку веб-додатків. Ці технології дозволяють створювати потужні та гнучкі додатки, які відповідають сучасним вимогам користувачів та бізнесу.

Основи роботи з Vue.js включають в себе розуміння його ключових концепцій та компонентів, що забезпечують гнучкість і ефективність розробки сучасних веб-додатків. Vue.js є прогресивним JavaScript-фреймворком, що дозволяє будувати інтерактивні інтерфейси користувача. Його основною перевагою є реактивність, яка автоматично відслідковує зміни в даних і оновлює DOM відповідно до цих змін.

Початковий крок у роботі з Vue.js — це створення нового Vue-екземпляра, який містить початкові дані та методи. Це можна зробити за допомогою конструктора `new Vue()`, де передається об'єкт конфігурації. У цьому об'єкті визначаються такі властивості як `el` (елемент DOM, до якого буде прив'язаний Vue-екземпляр), `data` (початкові дані), `methods` (методи, які можна використовувати в шаблоні) та `computed` (властивості, які обчислюються на основі інших властивостей).

Однією з основних концепцій Vue.js є використання шаблонів, які дозволяють визначати структуру HTML з інтеграцією динамічних даних. У шаблонах можна використовувати спеціальні директиви, такі як `v-bind` для прив'язки атрибутів, `v-model` для двостороннього зв'язку даних і `v-for` для ітерації по масивах.

Компоненти є ще однією ключовою частиною Vue.js. Вони дозволяють розбивати застосунок на незалежні, повторно використовувані частини. Кожен компонент має свій власний шаблон, логіку та стиль. Компоненти можуть бути як глобальними, так і локальними, і вони можуть взаємодіяти один з одним через властивості (props) та події (events).

Для керування станом у великих додатках використовується Vuex — офіційна бібліотека для керування станом у Vue.js. Vuex надає централізоване сховище для всіх компонентів додатку, що робить управління станом більш передбачуваним і простим.

Маршрутизація в Vue.js здійснюється за допомогою Vue Router, який дозволяє створювати односторінкові додатки з декількома маршрутами. Vue Router забезпечує динамічну навігацію та завантаження компонентів у відповідь на зміни URL.

Для збірки та оптимізації додатків Vue.js використовується Webpack, який дозволяє збирати модулі, мінімізувати файли та виконувати інші оптимізації. Vue CLI надає інструменти для швидкого створення та налаштування проєктів Vue.js з інтеграцією Webpack та інших інструментів.

Таким чином, основи роботи з Vue.js включають створення Vue-екземплярів, використання шаблонів і директив, розробку компонентів, керування станом з Vuex, маршрутизацію з Vue Router та оптимізацію проєктів з Webpack. Ці концепції та інструменти забезпечують розробникам ефективність та гнучкість у створенні сучасних інтерактивних веб-додатків.

1.3 Архітектура, компоненти, реактивність та управління станом у VUE

Архітектура та компоненти Vue.js базуються на принципах реактивного програмування та компонентного підходу, що забезпечують гнучкість і масштабованість у розробці сучасних веб-додатків.

Основою Vue.js є реактивна система, яка дозволяє автоматично відслідковувати зміни стану даних і оновлювати DOM відповідно до цих змін. Кожен Vue-екземпляр містить початкові дані (`data`), методи (`methods`) та обчислювані властивості (`computed`), які взаємодіють з шаблоном для динамічного рендерингу інтерфейсу.

Компоненти є ключовими елементами архітектури Vue.js. Вони дозволяють розбивати застосунок на незалежні, повторно використовувані частини. Кожен компонент має свій власний шаблон, логіку та стиль, що робить їх автономними та модульними. Компоненти можна створювати як глобально, так і локально. Глобальні компоненти реєструються у кореневому екземплярі Vue, тоді як локальні компоненти визначаються у межах інших компонентів.

Шаблони у Vue.js використовують спеціальні директиви, такі як `v-bind` для прив'язки атрибутів до даних, `v-model` для двостороннього зв'язку даних та `v-for` для ітерації по масивах. Директиви дозволяють легко інтегрувати динамічні дані у статичну HTML-структуру.

Для керування станом у великих додатках використовується Vuex, який надає централізоване сховище для всіх компонентів додатку. Vuex забезпечує передбачуваність та прозорість у керуванні станом, дозволяючи зберігати всі дані в одному місці та контролювати їх зміни через строго визначені дії (actions) та мутації (mutations).

Маршрутизація у Vue.js реалізується за допомогою Vue Router, який дозволяє створювати односторінкові додатки з динамічними маршрутами. Vue Router забезпечує легке керування навігацією, підтримуючи як прості маршрути, так і складні, що включають вкладені маршрути та асинхронне завантаження компонентів.

Vue.js також підтримує розширення функціональності через плагіни, які можуть додавати глобальні методи та властивості, а також додаткові директиви та фільтри. Це дозволяє легко інтегрувати сторонні бібліотеки та розширення у Vue-додатки.

Для збірки та оптимізації додатків Vue.js використовується Webpack, який дозволяє збирати модулі, мінімізувати файли та виконувати інші оптимізації. Vue CLI надає інструменти для швидкого створення та налаштування проєктів Vue.js з інтеграцією Webpack та інших інструментів, що спрощує розробку та розгортання додатків.

Таким чином, архітектура та компоненти Vue.js забезпечують гнучкість, масштабованість та ефективність у розробці веб-додатків. Компонентний підхід, реактивна система, централізоване керування станом з Vuex та маршрутизація з Vue Router створюють основу для побудови сучасних, динамічних та інтерактивних інтерфейсів користувача.

Реактивність та управління станом в Vue.js представляють собою ключові концепції, які забезпечують ефективність та зручність у розробці веб-додатків. Реактивність в Vue.js означає автоматичне відслідковування змін у стані даних та автоматичне оновлення відповідних частин інтерфейсу користувача без необхідності явного втручання програміста.

Основою реактивності в Vue.js є система об'єктів Vue, які спостерігають за даними, використовуючи реактивні властивості ES6 Proxy або Object.defineProperty, в залежності від доступності браузера. Коли дані змінюються, Vue автоматично визначає, які частини DOM потребують оновлення і виконує ці оновлення ефективно.

Управління станом в Vue.js зазвичай здійснюється через централізовані сховища, такі як Vuex. Vuex — це бібліотека управління станом, яка забезпечує однозначність та передбачуваність в управлінні станом за допомогою шаблону Flux. Вона включає в себе централізоване сховище, дії для зміни стану (actions), мутації для модифікації стану (mutations) та геттери для отримання даних зі сховища.

Vue також підтримує обчислювані властивості (computed properties) і спеціальні спостереження (watchers), які дозволяють реагувати на зміни в даних і виконувати відповідні обчислення або дії. Це дозволяє розробникам створювати складні логіку інтерфейсів, яка автоматично оновлюється при зміні вхідних даних.

Загальною метою реактивності та управління станом в Vue.js є забезпечення простоти, прозорості і ефективності у взаємодії з даними та відображенням користувацького інтерфейсу, що робить Vue.js потужним інструментом для розробки сучасних веб-додатків.

1.4 Інтеграція VUE з іншими бібліотеками та фреймворками та основи роботи

Інтеграція Vue.js з іншими бібліотеками та фреймворками відкриває широкі можливості для розширення функціональності і покращення розробки веб-додатків. Vue.js легко інтегрується з такими популярними бібліотеками та фреймворками, як:

Vuex - це бібліотека для керування станом, спеціально розроблена для Vue.js. Vuex надає централізоване сховище для всіх компонентів додатку та забезпечує однозначність управління станом.

Vue Router - офіційний маршрутизатор Vue.js для створення односторінкових додатків з динамічними маршрутами. Vue Router дозволяє легко налаштовувати маршрути в Vue-додатках і керувати навігацією.

Axios - це популярна бібліотека для виконання HTTP-запитів з JavaScript. Axios легко інтегрується з Vue.js для виконання AJAX-запитів та обміну даними з сервером.

Element UI / Vuetify- це UI-фреймворки, побудовані на основі Vue.js, які надають готові компоненти і стилі для швидкої розробки інтерфейсу користувача. Вони ідеально інтегруються з Vue.js, дозволяючи швидко створювати стильні та функціональні UI-елементи.

Chart.js / D3.js - бібліотеки для візуалізації даних, які можна легко інтегрувати з Vue.js для створення інтерактивних графіків та діаграм.

Firebase / AWS Amplify - платформи для хмарного обчислення, які надають послуги з аутентифікації, зберігання даних, реального часу та інші. Вони можуть бути легко інтегровані з Vue.js за допомогою спеціальних бібліотек і SDK.

Інтеграція з іншими бібліотеками та фреймворками дозволяє розширювати функціональність Vue.js, додавати нові можливості та швидко розробляти складні додатки з урахуванням сучасних вимог до веб-розробки.

Vue.js забезпечує гнучкість та простоту інтеграції, що робить його популярним вибором для розробників усього світу.

Laravel є потужним PHP-фреймворком, створеним Тейлором Отвеллом у 2011 році. Він відомий своїм зручним синтаксисом та широким набором інструментів для швидкої та ефективної розробки веб-додатків. Основні переваги Laravel включають простоту використання, потужні функціональні можливості та активну спільноту розробників[5].

У випадку з Laravel для бекенду інтеграція з іншими бібліотеками і фреймворками може включати розширення функціональності через пакети Composer, наприклад, для автентифікації, обробки платежів або взаємодії з базами даних. Також можна використовувати фронтенд фреймворки, такі як Vue.js, у поєднанні з Laravel за допомогою API, що дозволяє створювати потужні односторінкові додатки (SPA).

Для початку роботи з Laravel необхідно встановити Composer – менеджер залежностей для PHP. За допомогою Composer ви можете швидко створити новий проєкт Laravel командою `composer global require laravel/installer`. Далі створюється новий проєкт за допомогою команди `laravel new my-project`, що автоматично створює базову структуру проєкту з необхідними файлами та конфігураціями[6].

Основні компоненти Laravel включають маршрутизацію, яка відповідає за управління запитами та визначення шляхів у додатку. Маршрути визначаються у файлі `routes/web.php`. Контролери відповідають за обробку бізнес-логіки та створюються за допомогою Artisan-команди `php artisan make:controller HomeController`. Моделі та ORM Eloquent забезпечують взаємодію з базою даних і створюються командою `php artisan make:model User`. Міграції, які управляють структурою бази даних, створюються командою `php artisan make:migration create_users_table`.

Щоб створити перший застосунок на Laravel, необхідно визначити маршрут та створити контролер. Наприклад, маршрут може мати вигляд

`Route::get('/hello', 'HelloController@index')`, а контролер, який обробляє цей маршрут, може виглядати так:

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HelloController extends Controller
{
    public function index()
    {
        return 'Hello, Laravel!';
    }
}
```

Рисунок. 1.1 – Приклад використання маршрутів та контролерів

Це простий приклад того, як використовувати маршрути та контролери для створення додатку на Laravel(рис. 1.1).

MVC (Model-View-Controller) архітектура

LARAVEL використовує архітектурний патерн MVC, що забезпечує чітке розділення логіки додатку на три компоненти: модель, представлення та контролер.

Моделі відповідають за роботу з даними. В LARAVEL моделі базуються на ORM Eloquent, що забезпечує простий та інтуїтивний інтерфейс для взаємодії з базою даних.

Контролери обробляють HTTP-запити та повертають відповідні відповіді. Контролери керують логікою додатку та взаємодіють з моделями для отримання даних.

Представлення (Views) відповідають за відображення даних користувачам. Представлення створюються з використанням шаблонів Blade.

У великих проєктах важливо правильно організувати код для забезпечення його підтримуваності та розширюваності. Структура файлів у

Laravel організована згідно з сучасними стандартами веб-розробки і дозволяє краще управляти проектом. Основні каталоги включають кореневий каталог з конфігураційними файлами, такими як `.env` для змінних середовища і `composer.json` для залежностей Composer. В каталозі `app` розташована бізнес-логіка, включаючи моделі, контролери, middleware і ресурси. Каталоги `bootstrap` і `config` містять файли для завантаження середовища і конфігураційні файли відповідно. `Database` містить міграції, фабрики тестування і насіння, а `public` — веб-корінь з вхідним файлом `index.php` для обробки HTTP-запитів. В `resources` знаходяться необроблені ресурси і шаблони, а в `routes` — всі маршрути додатка. Каталог `storage` використовується для зберігання завантажених файлів, сесій, кешу і логів. `Tests` містить автоматизовані тести, а користувацькі файли можна створювати для специфічних потреб проекту.

Підключення до бази даних в Laravel налаштовується у файлі `.env`, де вказуються параметри з'єднання, такі як тип бази даних, хост, порт, назва бази даних, а також ім'я користувача та пароль для доступу до неї.

Міграції в Laravel дозволяють створювати та керувати структурою бази даних з допомогою PHP-коду. Це дозволяє створювати таблиці та модифікувати їх безпосередньо з коду за допомогою міграційних файлів. Після створення міграції її необхідно виконати командою `php artisan migrate`.

Eloquent ORM в Laravel забезпечує зручний інтерфейс для взаємодії з базою даних. Моделі Eloquent дозволяють легко створювати, зчитувати, оновлювати та видаляти записи в базі даних. Клас моделі визначає таблицю, з якою вона пов'язана, та масив `$fillable`, що містить поля, які можуть бути заповнені масово.

Запити до бази даних з використанням Eloquent прості та інтуїтивно зрозумілі. Вони дозволяють виконувати операції, такі як вибірка записів за умовою або по ID, а також комплексні запити з використанням різних методів Eloquent.

Ці основні аспекти дозволяють розробникам зручно та ефективно взаємодіяти з базами даних в Laravel, спрощуючи процес розробки та забезпечуючи високий рівень безпеки та ефективності управління даними.

Laravel надає вбудовані засоби для реалізації аутентифікації користувачів. Для встановлення аутентифікаційного пакету потрібно виконати команду `composer require laravel/ui`, а потім `php artisan ui vue --auth`. Ці команди автоматично створюють усі необхідні маршрути, контролери та представлення для реалізації аутентифікації.

Laravel також надає зручні засоби для авторизації та контролю доступу, такі як Gate та Policy. Наприклад, Gate дозволяє визначити правила доступу, які використовуються для перевірки прав користувачів на виконання певних дій у додатку.

Щодо безпеки, Laravel автоматично захищає застосунок від CSRF (міжсайтових запитів з підробленими запитами) за допомогою токенів, які додаються до всіх форм. Крім цього, він надає вбудовані механізми для захисту від XSS (міжсайтових скриптових атак), шляхом автоматичного екранування виведення даних у шаблонах Blade, що дозволяє попередити впровадження шкідливого коду на сторінках веб-додатку.

Laravel забезпечує зручні засоби для інтеграції з Vue.js. Для початку інтеграції необхідно встановити необхідні пакети, використовуючи команду `composer require laravel/ui`. Потім потрібно ініціалізувати Vue.js, використовуючи `php artisan ui vue`, і налаштувати Webpack, виконавши команди `npm install` та `npm run dev`.

Створення односторінкового додатка (SPA) з використанням Vue та Laravel дозволяє підвищити зручність використання та продуктивність. Налаштування SPA включає створення маршрутів Vue Router для компонентів, таких як Home і About, і підключення їх до основного шаблону додатка.

Проаналізовано можливості та функціональні характеристики фреймворків Vue.js та Laravel, які використовуються для розробки сучасних веб-додатків. Визначено, що Vue.js є потужним інструментом для створення

інтерактивного і динамічного інтерфейсу користувача, завдяки своїй гнучкості, реактативності та простоті в інтеграції з іншими бібліотеками. З іншого боку, Laravel, як бекенд фреймворк, пропонує багатий набір інструментів для роботи з базою даних, управління маршрутами, аутентифікації користувачів та забезпечення безпеки додатку. З’ясовано, що інтеграція цих двох фреймворків дозволяє досягти високого рівня продуктивності, масштабованості та зручності в розробці та підтримці веб-додатків. Проведений аналіз також показав, що використання Vue.js та Laravel значно спрощує процес розробки, скорочуючи час на реалізацію основних функціональних можливостей, що є важливим фактором для сучасних динамічних проектів.

2. РОЗРОБКА ВЕБ-ДОДАТКУ З ВИКОРИСТАННЯМ VUE ТА LARAVEL

2.1 Постановка задачі та порівняння з іншими аналогами

Розробка сучасного веб-додатку вимагає використання ефективних інструментів для фронтенду та бекенду. Фреймворки VUE.js та Laravel є потужними засобами для створення масштабованих і продуктивних веб-додатків.

Постановка завдання для дипломної роботи полягає в розробці онлайн платформи для курсів та навчання, використовуючи фронтенд на Vue.js і бекенд на Laravel. Основні цілі проєкту включають дослідження можливостей обох фреймворків, їх інтеграцію та оцінку їх функціональних можливостей і обмежень. Аналіз потреб користувачів, розробка зручного і функціонального інтерфейсу для користувачів і адміністраторів, забезпечення безпеки і швидкості роботи системи - це ключові аспекти, які будуть враховані під час розробки. Також в рамках завдання буде вирішено питання авторизації користувачів, можливість створення та керування курсами, систему підписки та інші функціональні можливості, що відповідають сучасним стандартам веб-розробки і очікуванням користувачів.

Почав з аналізу потреб цільової аудиторії - студентів, викладачів, адміністраторів тощо. Визначив їх вимоги та очікування від платформи курсів та навчання. Встановив основні цілі та завдання проєкту, які включають створення зручного середовища для навчання, доступ до різноманітних курсів та матеріалів, можливість спілкування та співпраці користувачів тощо. Визначив перелік основних функцій та можливостей, які має мати платформа. Це включає реєстрацію та авторизацію користувачів, створення та відображення курсів, можливість завантаження матеріалів, форум для обговорень, оцінювання та відстеження прогресу навчання тощо.

Визначив пріоритети для реалізації різних функцій та можливостей в рамках проєкту. Це допомогло сконцентруватися на найважливіших аспектах платформи та етапувати розробку. Встановив технічні вимоги до системи, включаючи вибір технологій, архітектуру додатку, потреби у масштабуванні та безпеці.

Розробив план дій для реалізації поставлених цілей та завдань, включаючи розподіл робіт між командою розробників, установку термінів та контроль робочого процесу.

Аналізуючи фреймворки VUE.js та Laravel, я вивчив їх основні характеристики, архітектуру та інструменти. VUE.js вражає своєю простотою та ефективністю у створенні інтерактивних користувацьких інтерфейсів, що базуються на компонентах. Laravel, у свою чергу, відзначається потужним ORM для роботи з базою даних, ефективним роутингом та зручним шаблонізатором Blade.

Завдяки отриманій інформації я розробив веб-застосуно, який інтегрує VUE.js для фронтенду та Laravel для бекенду. Впевнився в успішній інтеграції між цими двома фреймворками, забезпечивши повний цикл обробки даних від користувача до сервера і назад. Після розробки я провів тестування додатку, оцінивши його продуктивність і зручність використання. Порівняв VUE.js та Laravel з іншими популярними фреймворками, виокремивши їхні переваги та недоліки.

У результаті роботи отримав функціональний веб-застосунок, який демонструє інтеграцію VUE.js та Laravel, а також надав рекомендації щодо їх використання у майбутніх проєктах. Застосунок включає інтерактивний користувацький інтерфейс, централізоване управління станом, аутентифікацію користувачів і операції CRUD з базою даних.

Порівняння моєї платформи курсів та навчання онлайн з іншими аналогами може бути корисним для користувачів, які розглядають різні варіанти перед прийняттям рішення. Ось деякі аспекти, за якими можна провести порівняння:

Моя платформа пропонує широкий вибір курсів з різних галузей знань, включаючи IT, бізнес, мистецтво та інші. Порівняно з іншими платформами, ми можемо мати більш широкий асортимент навчальних програм. Ми зосереджуємося на якості навчання, співпрацюючи з висококваліфікованими викладачами та надаючи доступ до якісних навчальних матеріалів. Це може відрізнятися від інших платформ, де якість контенту може бути менш консистентною. Інтерактивність та залучення: Наша платформа ставить на активне залучення студентів через інтерактивні вправи, форуми обговорень та інші інструменти. Порівняно з більш традиційними платформами, ми можемо надати більші можливості для активного навчання. Порівнюючи з іншими платформами, ми можемо пропонувати конкурентні ціни та можливості для отримання доступу до безкоштовних або дешевих курсів. Це може зробити нашу платформу більш доступною для широкого кола користувачів.

Ефективна система підтримки користувачів та активна спільнота, яка може допомогти студентам вирішити будь-які питання чи проблеми, які вони можуть мати. Це може бути важливим фактором при виборі платформи для навчання. Ці аспекти можуть допомогти користувачам зробити більш обґрунтований вибір при виборі платформи курсів та навчання онлайн.

2.2 Проєктування архітектури додатку

У процесі проєктування архітектури моєї онлайн платформи курсів та навчання онлайн я врахував ряд ключових аспектів, щоб забезпечити ефективну та масштабовану систему. Ось кроки, які я виконав:

Почав з вивчення потреб цільової аудиторії та визначення їх вимог до платформи. Це дозволило мені зрозуміти, які функціональність та можливості повинен мати мій застосунок. Визначив структуру додатку, включаючи головні функціональні модулі та їх взаємодію. Розробив схему навігації та визначив основні компоненти і елементи інтерфейсу. Обрав набір технологій, який відповідає потребам мого додатку. Врахував фронтендні та бекендні

технології, базу даних та інші компоненти. Вирішив використати реляційну або NoSQL базу даних для зберігання інформації про курси, користувачів та інші дані. Врахував потреби у швидкодії, масштабованості та безпеці. Використовував архітектурні шаблони, такі як MVC або MVVM, для розділення логіки додатку та його компонентів. Це дозволило забезпечити структурованість та легкість управління кодом. Розробив архітектуру з урахуванням можливості масштабування та надійності. Врахував високу навантаженість та обробку одночасних запитів.

Проєктування архітектури додатку включає розробку структурних рішень, що забезпечують ефективну взаємодію між компонентами фронтенду та бекенду, а також дотримання принципів масштабованості, модульності та підтримуваності.

Фронтенд частину мого додатку я розробив з використанням Vue.js. Я використовував компонентну структуру для побудови інтерфейсу користувача, що дозволяє ефективно організувати код, забезпечуючи модульність і можливість повторного використання компонентів. Для управління маршрутами і навігацією між сторінками я використовував Vue Router, що дозволило зручно налаштовувати маршрути в додатку.

Управління станом фронтенду було організоване за допомогою Vuex, що забезпечує централізоване управління станом додатку і спрощує роботу зі змінними даними на всій його площині.

Щодо бекенду, я використовував фреймворк Laravel. Я застосовував MVC-патерн для чіткого розмежування логіки додатку на моделі, представлення та контролери. Це дозволяє ефективно управляти функціональними частинами додатку та забезпечувати його розширюваність.

Для забезпечення взаємодії фронтенду із серверною частиною я розробив RESTful API, що дозволяє передавати дані між клієнтом і сервером у зручному форматі.

Окрім цього, для аутентифікації та авторизації користувачів я використовував вбудовані можливості Laravel, що дозволило забезпечити безпеку та доступ до ресурсів додатку.

Цей підхід дозволив мені ефективно інтегрувати фронтенд із бекендом, створюючи функціональний веб-застосунок з дотриманням сучасних стандартів і найкращих практик розробки.

Проєктування архітектури додатку з використанням VUE.js та Laravel забезпечило чітке розмежування між фронтендом і бекендом, сприяло модульності та підтримуваності коду, а також дозволило ефективно масштабувати застосунок відповідно до зростаючих вимог. На фронтенді використовувались компоненти VUE, Vue Router для налаштування маршрутів і навігації в додатку, а також Vuex для централізованого управління станом додатку. Бекенд був реалізований з використанням фреймворку Laravel, використовуючи MVC-паттерн для чіткого розмежування бізнес-логіки на моделі, представленні та контролерах. Для роботи з базою даних використовувався Eloquent ORM, що спрощує взаємодію з MySQL і забезпечує швидкий доступ до даних.

Аутентифікація користувачів була реалізована за допомогою вбудованих можливостей Laravel, що включають реєстрацію та авторизацію користувачів, забезпечуючи безпеку та доступ до особистих облікових записів. Ця архітектурна модель дозволила ефективно інтегрувати фронтенд і бекенд, створюючи функціональний та високопродуктивний веб-застосунок, готовий відповідати вимогам сучасного ринку і задовольняти потреби користувачів.

2.3 Розробка бекенд та фронтенд-частини з використанням VUE та LARAVEL

Після створення нового проєкту Laravel потрібно налаштувати підключення до бази даних у файлі `.env`, вказавши параметри для з'єднання з MySQL: хост (`DB_HOST`), порт (`DB_PORT`), назву бази даних

(`DB_DATABASE`), користувача (`DB_USERNAME`) та його пароль (`DB_PASSWORD`) [10].

Для створення моделей і відповідних міграцій використовується Artisan, команда `php artisan make:model Post -m` автоматично створить модель `Post` і відповідну міграцію для таблиці `posts`, де можна визначити поля таблиці і їх типи даних.

Міграції виконуються командою `php artisan migrate`, що створює таблицю в базі даних відповідно до описаної схеми у міграційному файлі.

Для обробки CRUD операцій з постами створюється контролер `PostController` за допомогою команди `php artisan make:controller PostController --resource`. Цей контролер міститиме методи для відображення всіх постів, створення нового поста, відображення конкретного поста за його ідентифікатором, оновлення і видалення поста.

Маршрути для контролера `PostController` визначаються у файлі `routes/web.php` за допомогою методу `Route::resource('posts', PostController::class)`, що автоматично створює необхідні маршрути для кожного з методів контролера.

Цей підхід дозволяє ефективно організувати роботу з базою даних і керування даними у веб-додатку, забезпечуючи структурованість та чіткість коду.

Для розробки фронтенд-частини за допомогою Vue.js я використовував компонентний підхід для побудови інтерфейсу користувача. За допомогою Vue компонентів я забезпечив модульність і багаторазове використання коду, що спростило процес розробки та підтримки додатку.

Для керування маршрутами та навігацією я використовував Vue Router. Це дозволило ефективно організувати переходи між сторінками додатку і забезпечити зручну навігацію для користувачів.

Управління станом фронтенду було забезпечено за допомогою Vuex. Я використовував цю бібліотеку для централізованого зберігання даних і

управління станом додатку. Це дозволило ефективно керувати станом компонентів і забезпечити консистентність даних на всій платформі.

Для взаємодії з сервером і обробки даних я використовував Axios, що забезпечило зручний і надійний спосіб відправки запитів до API бекенду та обробки отриманих відповідей [11].

2.4 Вигляд веб-додатку та рекомендації з використання

Під-час створення додатку було створено поле реєстрації з можливістю увійти і зареєструватися у свій обліковий запис або створити новий обліковий запис, щоб почати. Збереження вашої особистої інформації в безпеці за допомогою системи автентифікації (рис. 2.1).

Рисунок 2.1 - Поле реєстрації

Керування профілем в моєму веб-додатку навчання онлайн дозволяє користувачам керувати своїми особистими даними, налаштуваннями і взаємодіяти з платформою. Основні функції керування профілем можуть включати: редагування особистої інформації, перегляд курсів, редагування паролів (рис. 2.2).

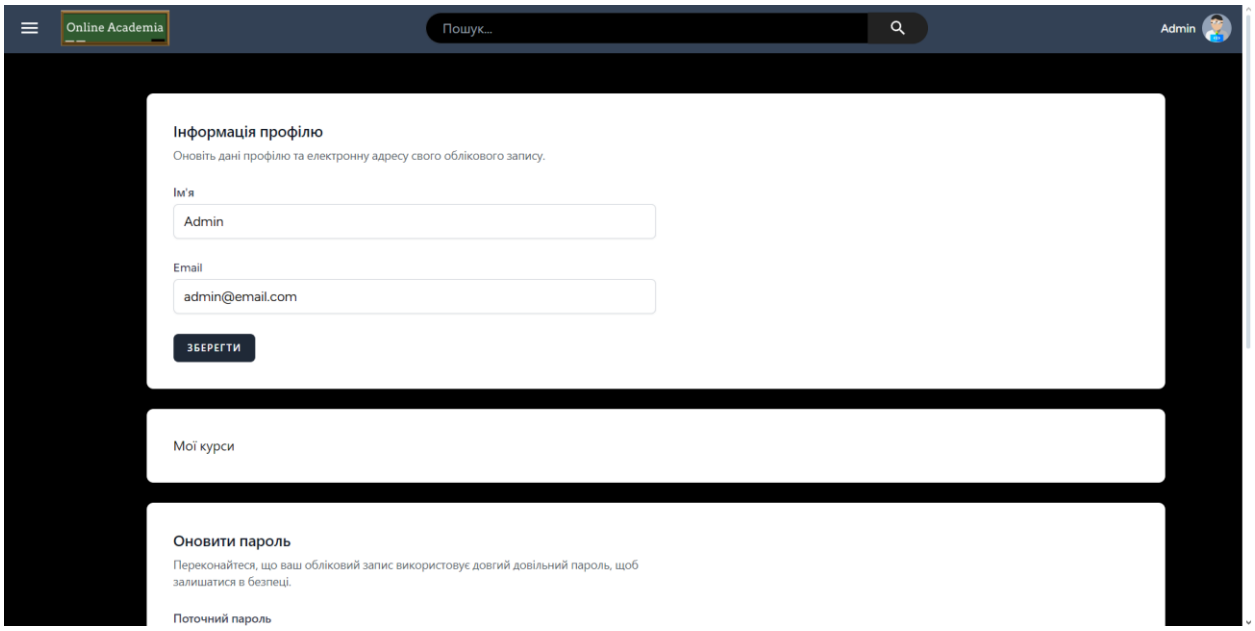

Рисунок 2.2 - Керування профілем

Меню курсів у веб-додатку навчання онлайн відіграє ключову роль у зручному навігаційному досвіді користувачів. Воно пропонує широкий вибір навчальних програм, які розділені на категорії за темами, наприклад програмування, мови, бізнес або мистецтво. Кожен курс має докладний опис, що включає інформацію про програму навчання, матеріали, що використовуються, і відгуки користувачів. Користувачі можуть використовувати фільтри і пошукові інструменти для швидкого знаходження курсів за інтересами та потребами. Рейтинги та відгуки дозволяють зробити обґрунтований вибір, а додаткові функції, такі як зберігання улюблених курсів і отримання рекомендацій, покращують загальний досвід користувачів у процесі навчання (рис. 2.3).

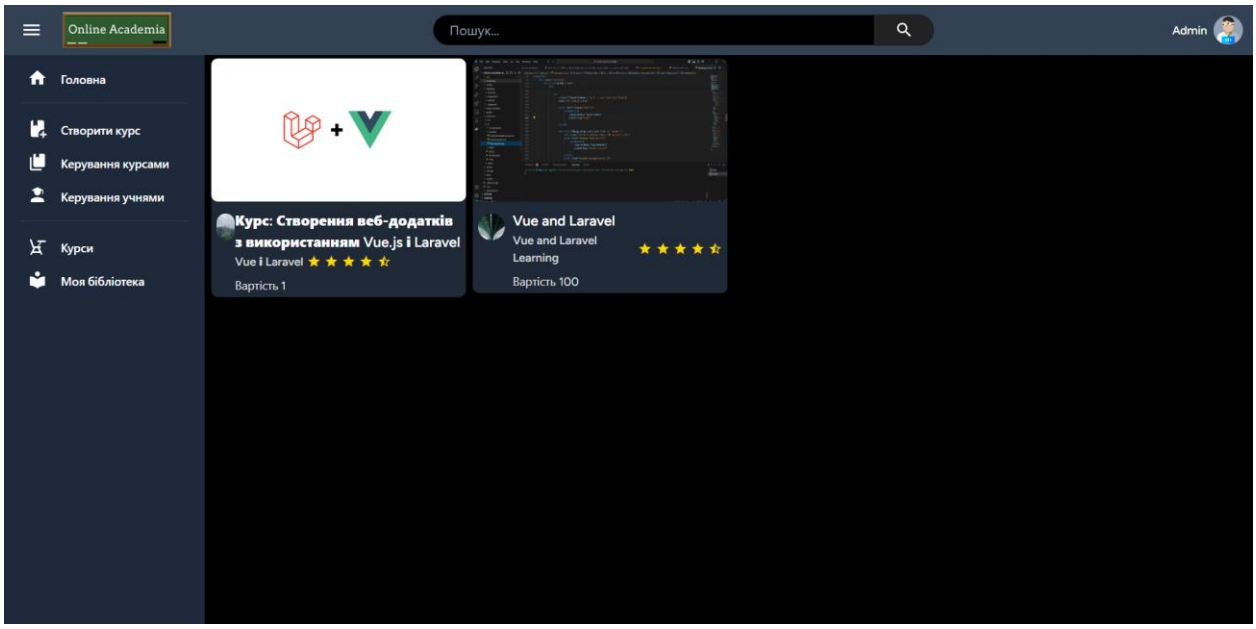

Рисунок 2.3 - Меню курсів

Меню додавання курсів у веб-додатку навчання онлайн є інструментом для викладачів і адміністраторів, який дозволяє легко і ефективно створювати нові навчальні програми та додавати їх до платформи. Це меню надає можливість заповнити всю необхідну інформацію про курс, включаючи його назву, опис, програму навчання і вимоги до учасників. Крім того, викладачі можуть завантажувати різноманітні матеріали для кожного уроку або модуля, такі як відеоуроки, тексти, завдання, тести та інші ресурси, необхідні для ефективного навчання. Вони також мають можливість налаштувати параметри курсу, такі як тривалість, рівень складності і категорія, що допомагає користувачам знайти та вибрати курс, що відповідає їхнім потребам. Після створення курс може бути перевірений і опублікований з метою доступу для учасників, і відслідковувати їх прогрес і результати через інструменти аналітики (рис. 2.4).

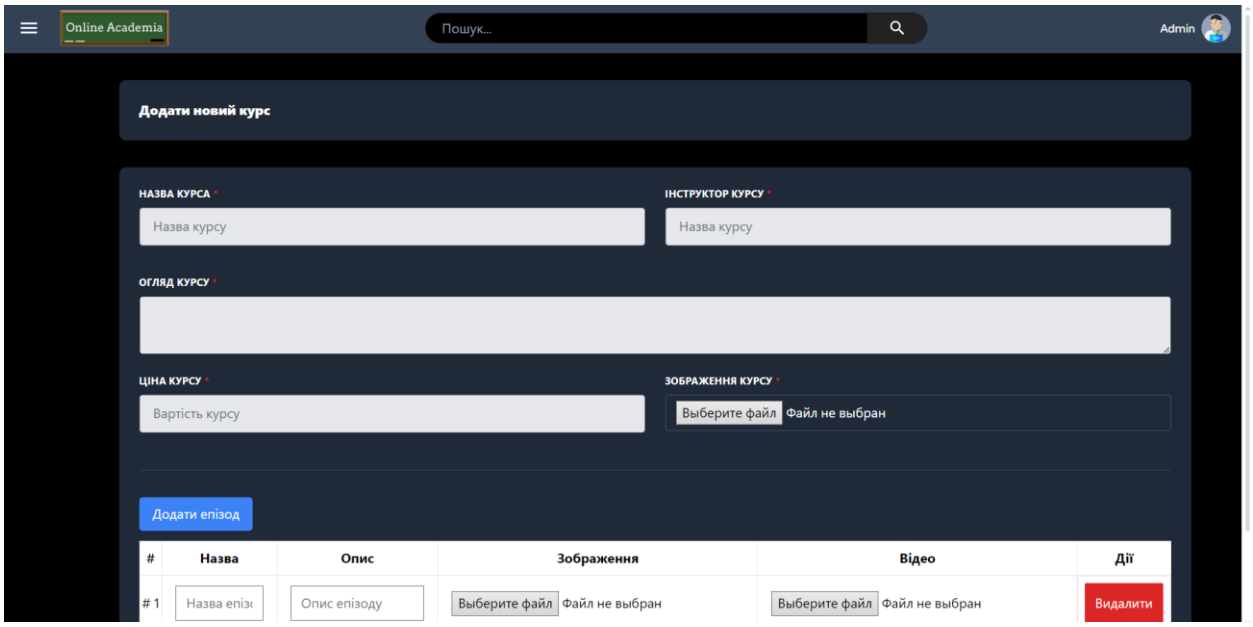

Рисунок 2.4 - Меню додавання курсів

Керування та видалення курсів у веб-додатку є важливими аспектами для забезпечення актуальності і якості навчальної інформації. Адміністратори і викладачі мають можливість легко керувати курсами, змінювати і оновлювати їх інформацію, додавати нові матеріали і налаштовувати параметри доступу. Вони також можуть видаляти курси з платформи після підтвердження, забезпечуючи видалення курсу безпечно і без втрати даних. Управління курсами включає в себе можливість відновлення видалених курсів за необхідності і збирання аналітики щодо видалених програм для подальшого вдосконалення управлінських рішень. Ці процеси дозволяють забезпечити користувачам платформи доступ до актуальної та оновлюваної навчальної інформації, що відповідає їх потребам і покращує їхній досвід навчання онлайн (рис. 2.5).

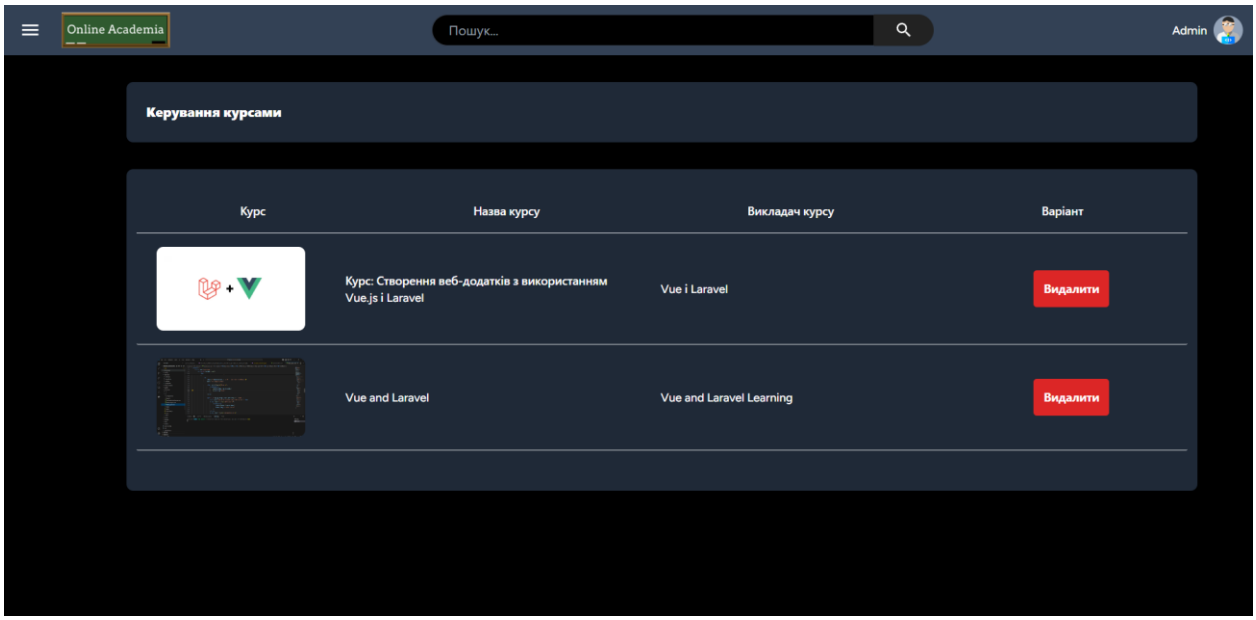

Рисунок 2.5 - Меню керування та видалення курсів

Перегляд сторінки курсу у веб-додатку навчання онлайн є ключовою можливістю для користувачів, які бажають отримати детальну інформацію про конкретну навчальну програму перед тим, як зареєструватися на нього чи почати навчання. Ця сторінка зазвичай містить повний опис курсу, включаючи його мету, основні теми, які будуть вивчатися, і загальний перелік уроків або модулів. Користувачі можуть ознайомитися з програмою курсу, переглянути матеріали, які включаються до навчального процесу, а також визначити вимоги до учасників, необхідні для успішного проходження. Крім того, сторінка курсу може включати інформацію про авторів або викладачів курсу, їхні кваліфікації і досвід, що додає додаткову вірогідність інформації. Оцінки та відгуки інших користувачів також часто представлені на сторінці курсу, що дозволяє користувачам отримати об'єктивну оцінку якості і корисності курсу перед його вибором (рис. 2.6).

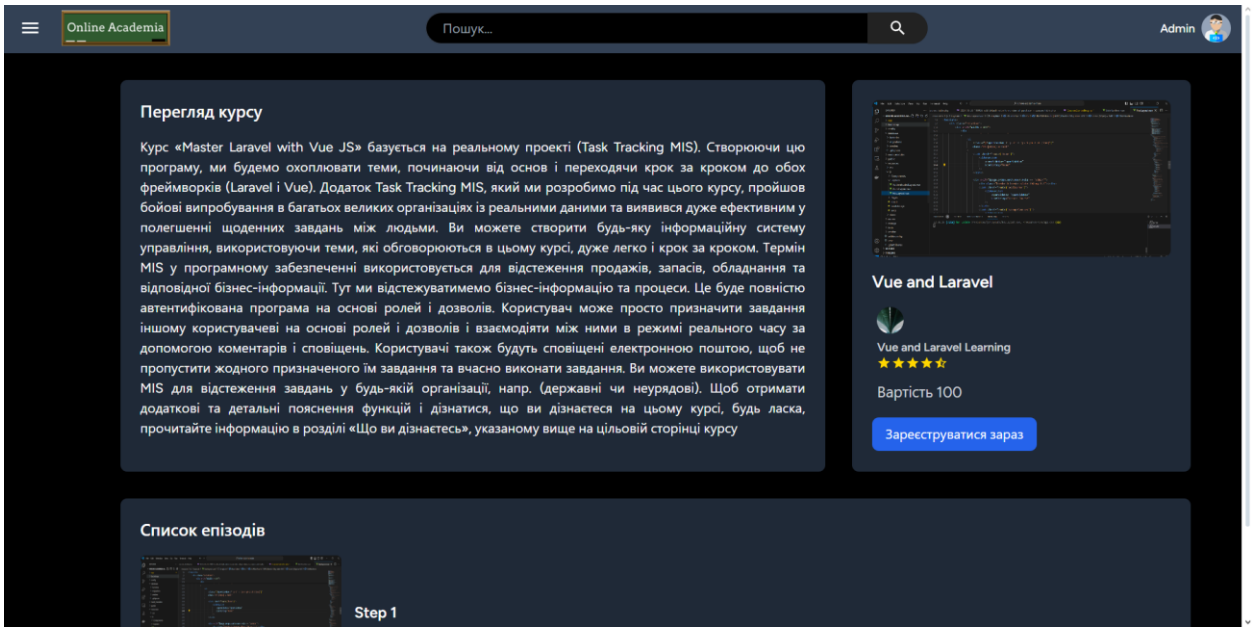

Рисунок 2.6 - Меню перегляду курсів

2.5 Тестування та налагодження додатку

Після завершення розробки я провів комплексне тестування та налагодження додатку з використанням Vue.js та Laravel. Для фронтенду з Vue.js я використовував юніт-тести для перевірки окремих компонентів за допомогою Jest та Vue Test Utils. Також були написані тести інтерфейсу для перевірки різних станів компонентів і енд-то-енд (E2E) тести з використанням Cypress для тестування повної функціональності додатку.

Для бекенду з Laravel я налаштував юніт-тести з використанням PHPUnit для перевірки окремих методів і функцій, а також функціональні тести для перевірки роботи конкретних функціональних частин, наприклад, API-запитів. Інтеграційні тести допомогли перевірити взаємодію між різними частинами системи, включаючи базу даних та зовнішні сервіси.

Для налагодження додатку я використовував консоль браузера для виявлення та виправлення помилок у JavaScript-коді, Vue Devtools для налагодження Vue.js додатків, логування Laravel для зберігання інформації про помилки та Xdebug для детального аналізу PHP-коду.

Також була налаштована безперервна інтеграція (CI) з використанням GitHub Actions для автоматичного виконання тестів при кожному оновленні коду у репозиторії, а безперервна доставка (CD) забезпечила автоматичне розгортання додатку після успішного завершення всіх тестів.

Такий підхід забезпечив високу якість коду, надійність і стабільність веб-додатку, зменшивши ризик виникнення помилок у продуктивному середовищі та покращивши ефективність розробки.

Після завершення розробки моєї веб-додатку, я ретельно пройшов через всі етапи тестування, щоб забезпечити високу якість і надійність продукту. План тестування включав кілька ключових етапів, що описуються нижче.

Налаштував тестове середовище для запуску веб-додатку і виконання тестів. Інсталював необхідні компоненти (сервер, база даних, залежності) для фронтенду та бекенду. Налаштував тестовий сервер та встановив фреймворки для автоматизації тестування.

Перевірив основні функції та операції веб-додатку. Виконав сценарії тестування, включаючи позитивні, негативні та граничні сценарії. Перевірив відповідність результатів очікуваним.

Перевірив коректність і зручність взаємодії з користувацьким інтерфейсом. Проводив тести на різних пристроях та браузерах для переконання у коректності відображення та функціональності.

Перевірив правильність обміну даними між фронтендом та бекендом через API. Впевнився в коректності формату запитів та відповідей, обробці помилок, а також забезпеченні безпеки даних.

Перевірив роботу веб-додатку на різних платформах та пристроях. Впевнився у сумісності з різними версіями браузерів та операційних систем.

Провів тести на вразливість, перевірив захист від SQL-ін'єкцій, XSS-атак та інших загроз. Аудитував код і застосовував правила безпеки для забезпечення надійності додатку.

Кожен етап тестування був виконаний послідовно з метою забезпечення високої якості та надійності веб-додатку, зменшення ризику виникнення помилок та покращення його ефективності.

Модульне тестування є важливою складовою процесу розробки мого веб-додатку, побудованого на фреймворках VUE.js та Laravel. Цей метод тестування дозволяє перевірити окремі компоненти програми на коректність роботи в ізоляції від інших частин системи. Основні кроки та підходи, які я використовував під час модульного тестування, включають:

Визначив конкретні функції, методи або класи, які підлягають модульному тестуванню. Це включало як фронтендові, так і бекендові компоненти додатку.

Розробив набір тестових сценаріїв для кожного обраного модуля. Тестові сценарії включали позитивні та негативні випадки, а також граничні умови для кожного методу або функції. Запустив розроблені тестові сценарії і проаналізував отримані результати. Впевнився, що кожен модуль працює згідно очікувань та відповідає специфікаціям. Виявлені під час тестування помилки були виправлені відповідно до вимог якості та стандартів коду. Після внесення виправлень повторно запустив тестові сценарії, щоб переконатися у коректності роботи виправлених модулів.

Очікування після проведення модульного тестування полягає в підтвердженні правильності роботи окремих компонентів програми та забезпеченні високої якості веб-додатку в цілому. Модульне тестування допомогло виявити та усунути помилки, що підвищило загальну надійність і ефективність мого додатку.

Тестування користувацького інтерфейсу включає в собі комплекс перевірок, що гарантують якість і зручність взаємодії користувача з моїм веб-додатком. Під час тестування я переконався у наступному:

Перш за все, було перевірено відображення всіх елементів інтерфейсу на різних пристроях і роздільних здатностях екрану, а також упевнився, що всі компоненти коректно відображаються і не перекривають один одного.

Навігаційні посилання та кнопки були протестовані на функціональність, і я впевнився, що користувач може легко переходити між різними сторінками та функціями додатку. Також було проведено тестування роботи всіх інтерактивних елементів, таких як кнопки, форми введення та меню, і переконався, що користувач може коректно взаємодіяти з усіма компонентами інтерфейсу.

Дизайн усіх елементів інтерфейсу було перевірено на відповідність специфікаціям та очікуванням користувачів. Я забезпечив, що дизайн додатку є привабливим і зручним для користувача. Тестування включало перевірку доступності всіх функцій та елементів інтерфейсу для користувачів з обмеженими можливостями, таких як люди з інвалідністю чи обмеженою моторикою.

Також було проведено тестування на різних браузерах і операційних системах, щоб підтвердити, що інтерфейс працює однаково добре на будь-яких пристроях. Адаптивність інтерфейсу до різних розмірів екранів було перевірено, забезпечуючи зручне користування незалежно від пристрою.

Також було здійснено перевірку коректності відображення текстів та елементів інтерфейсу для різних мов та культур. Ці кроки тестування забезпечили високу якість і зручність веб-додатку, що є критичним для першого враження користувача і загального успіху програмного забезпечення.

Виправлення помилок і вдосконалення системи є важливою частиною розробки програмного забезпечення, включаючи веб-додатки. Цей процес включає в себе кілька кроків, які допомагають поліпшити якість і функціональність системи.

Спочатку проводиться аналіз всіх виявлених помилок в системі, включаючи як функціональні, так і технічні недоліки. Потім помилки класифікуються за ступенем важливості та впливу на користувачів і бізнес-процеси. Важливі помилки, які можуть негативно позначитися на користувачах, виправлюються в першу чергу.

Розробники вносять відповідні зміни до коду системи для виправлення помилок. Після цього проводиться тестування, щоб переконатися, що виправлення були успішними і не виникли нові проблеми. Після внесення змін і виправлення помилок також проводиться регресійне тестування, щоб переконатися, що зміни не призвели до появи нових помилок у вже наявних функціях.

Необхідно оновити документацію після внесення змін у систему, щоб вона відображала актуальний стан системи і функціоналу. Після успішного виправлення помилок і вдосконалення системи зміни впроваджуються до виробничого середовища.

Після впровадження змін важливо відстежувати роботу системи і забезпечувати регулярну звітність про її стан, виявлені помилки і проведені вдосконалення. Цей процес є постійним і динамічним, оскільки розробники завжди шукають способи поліпшення якості і функціональності системи, щоб задовольнити потреби користувачів і вирішити бізнес-задачі.

Розробив веб-застосунок з використанням фреймворків Vue.js та Laravel, що дозволяє реалізувати всі основні функції платформи для онлайн-навчання. В процесі розробки були створені різноманітні компоненти користувацького інтерфейсу з Vue.js, які забезпечують інтуїтивно зрозумілу та зручну навігацію для користувачів. Застосування Laravel на серверній частині дозволило організувати ефективну роботу з базою даних, забезпечити безпеку та логіку обробки запитів. Було реалізовано механізм реєстрації та авторизації користувачів, управління курсами, а також додавання нових курсів та їх перегляд. Веб-додаток демонструє високу продуктивність, масштабованість та відповідає сучасним вимогам до веб-розробки. Розробка підтвердила доцільність використання Vue.js та Laravel для створення складних веб-додатків з багатим функціоналом.

3. ВПРОВАДЖЕННЯ СИСТЕМИ

3.1 Налаштування серверного середовища, розгортання бази даних

Налаштування серверного середовища є критично важливим етапом у розробці та впровадженні веб-додатків. Цей процес включає в себе ряд кроків, які спрямовані на забезпечення ефективності, безпеки та надійності сервера, на якому буде запущений ваш застосунок. Ось детальний огляд кроків, необхідних для належного налаштування серверного середовища:

Обрав сервер, який відповідає вимогам мого веб-додатку. Найпоширенішими варіантами є Apache, Nginx, а також спеціалізовані сервери, такі як Node.js для додатків на JavaScript.

Встановив необхідні компоненти, такі як веб-сервер, база даних, інтерпретатори мов програмування, розширення PHP та інше.

Налаштував конфігураційні файли веб-сервера (наприклад, httpd.conf для Apache або nginx.conf для Nginx) для встановлення параметрів, таких як порти, доменні імена, налаштування безпеки тощо. Встановив SSL-сертифікат для забезпечення захищеного з'єднання між клієнтом та сервером.

Створив базу даних та налаштував доступ до неї з веб-додатку. Впевнився, що налаштування бази даних відповідає вимогам мого додатку.

Провів аудит безпеки мого сервера та веб-додатку. Встановив правила файрволу, налаштував захист від атак, забезпечте безпеку доступу до сервера.

Налаштував систему моніторингу та логування для відстеження роботи сервера та виявлення можливих проблем. Виконав налаштування для оптимізації продуктивності сервера та веб-додатку, такі як кешування, стискання даних, налаштування пам'яті та інше.

Налаштував систему автоматичного резервного копіювання для захисту даних та можливості відновлення в разі аварій. Після аналізу різних можливостей для серверного середовища мого веб-додатку, я вирішив

використати хостинговий провайдер, який може забезпечити необхідні ресурси та зручний інтерфейс для налаштування. Обравши провайдера, я створив віртуальний сервер і налаштував його для оптимальної продуктивності та безпеки.

Далі я встановив та налаштував веб-сервер Nginx, оскільки він є швидким, надійним і має простий у використанні конфігураційний файл. Після цього я встановив та налаштував базу даних MySQL для зберігання всієї необхідної інформації про курси, користувачів та інше.

Особливу увагу приділив забезпеченню безпеки, встановивши сертифікат SSL для шифрування з'єднання і налаштувавши файрвол для захисту від несанкціонованого доступу. Крім того, налаштував систему регулярного резервного копіювання для забезпечення безпеки даних.

Після налаштування всіх компонентів серверного середовища я провів тестування на продуктивному середовищі, щоб переконатися, що все працює належним чином та відповідає потребам користувачів. У результаті, моє серверне середовище готове до роботи, забезпечуючи надійну та безпечну основу для моєї онлайн платформи курсів і навчання.

Під час розгортання бази даних для мого дипломного проекту я керувався наступними кроками: врахувавши вимоги проекту і мої власні навички, я обрав MySQL як систему керування базами даних. За допомогою документації MySQL та пакетного менеджера на моєму сервері встановив необхідну версію MySQL. За допомогою командного рядка MySQL створив нову базу даних з іменем "diploma_project_db" і розробив SQL скрипти для створення таблиць, які відповідають вимогам дипломної роботи. Додав початкові дані до таблиць бази даних для тестування та розробки. Створив окремого користувача бази даних з відповідними правами доступу і у файлі конфігурації додатку вказав параметри підключення до бази даних, такі як хост, ім'я бази даних, ім'я користувача та пароль. Перевірив, що з'єднання з базою даних працює, виконавши тестові запити з додатку. Налаштував автоматичне регулярне резервне копіювання бази даних для запобігання втраті

даних. Ці кроки дозволили успішно розгорнути базу даних для мого дипломного проекту і забезпечити стабільну та надійну роботу системи.

Після підготовки серверного середовища я приступив до розгортання бази даних для мого веб-застосунку. Обрав MySQL як систему керування базами даних, оскільки вона відома своєю потужністю та надійністю у веб-серверних застосунках.

Спочатку я створив нову базу даних в MySQL за допомогою команди `CREATE DATABASE`, обравши назву, що відповідає моїй веб-платформі курсів та навчання. Потім розробив структуру таблиць для зберігання інформації про користувачів, курси, матеріали та інші дані, використовуючи мову SQL та визначивши необхідні поля та типи даних.

Для перевірки роботи моєї веб-платформи заповнив таблиці тестовими даними, додавши кілька користувачів, курсів та інших елементів. Для забезпечення безпеки даних створив користувача бази даних та налаштував йому відповідні привілеї доступу, обмеживши лише до необхідних операцій.

Ці кроки дозволили успішно розгорнути базу даних для мого веб-застосунку. Тепер система готова забезпечувати надійне зберігання та доступ до даних, необхідних для роботи платформи.

3.2 Процес впровадження та оцінка ефективності системи

Процес впровадження моєї платформи курсів та навчання був детально спланований та етапований для забезпечення успішного запуску і максимальної ефективності. Перед впровадженням я здійснив підготовчі роботи, встановив необхідне обладнання, програмне забезпечення та створив інфраструктуру хостингу. Провів серію тестів для перевірки функціональності та стабільності платформи, виправивши виявлені помилки. Організував навчальні курси та тренінги для користувачів з використання нової платформи, включаючи онлайн-ресурси та навчальні сесії в реальному часі. Після успішного тестування та підготовки користувачів офіційно запустив

платформу і розпочав роботу в режимі реального часу. Продовжив моніторити роботу платформи та надавати підтримку користувачам, вирішуючи будь-які технічні або функціональні питання. Провів оцінку результатів впровадження, аналізуючи відгуки користувачів та реакцію на ринку, щоб визначити успішність процесу та ідентифікувати можливі шляхи вдосконалення. Цей процес впровадження дозволив успішно запустити мою платформу курсів та навчання та почати надання послуг користувачам у відповідь на їхні потреби та очікування.

Оцінка ефективності системи для моєї платформи курсів та навчання включала аналіз різних аспектів функціонування та використання платформи з точки зору користувачів та бізнес-цілей. Ось деякі з основних кроків, які я виконав під час оцінки ефективності системи:

Оцінка ефективності розпочалася зі збору даних про користувачів, їхні вимоги та побажання. Це включало аналіз даних про взаємодію користувачів з платформою, їхні відгуки та пропозиції.

Оцінив різні ключові метрики успішності, такі як кількість зареєстрованих користувачів, активність користувачів, кількість завершених курсів, час проведений на платформі тощо. Провів опитування та аналіз відгуків користувачів, щоб оцінити їхню задоволеність функціональністю, якістю навчального контенту, зручністю використання та іншими аспектами.

Порівняв результати з вихідними цілями та очікуваннями, щоб визначити, наскільки ефективно система відповідає бізнес-потребам та очікуванням користувачів. На основі отриманих результатів вніс необхідні зміни та вдосконалення у платформу, спрямовані на покращення її ефективності та задоволення потреб користувачів. Провів повторну оцінку системи через певний час, щоб перевірити ефективність внесених змін та визначити подальші напрямки розвитку.

3.3 Аналіз ринку та конкурентів

Проведення аналізу ринку та конкурентів для моєї онлайн платформи курсів та навчання було ключовим етапом у визначенні стратегії розвитку та позиціонування на ринку. Я розпочав зі збору інформації про різних конкурентів у сфері онлайн навчання, звертаючи увагу на відомі міжнародні платформи, такі як Coursera, Udemy, edX, Khan Academy, а також менш відомі локальні або спеціалізовані платформи[9]. Провів детальний аналіз їхньої функціональності та особливостей, оцінив переваги та недоліки кожної з них, а також якість навчального контенту, доступність курсів, інтерфейс користувача, ціноутворення, систему підтримки та інші важливі аспекти.

Детально вивчив ринкові тенденції та прогнози розвитку онлайн навчання, аналізуючи зростання популярності курсів у мережі, зміни в попиті на різні категорії курсів, тенденції в ціноутворенні та інші ключові фактори. На основі цього аналізу визначив конкурентні переваги моєї платформи, такі як унікальний контент, інноваційні методи навчання, гнучка система ціноутворення та висока якість обслуговування клієнтів.

Завершивши аналіз, розробив стратегію позиціонування моєї платформи на ринку, визначивши унікальні цільові аудиторії, ключові повідомлення та ефективні способи комунікації з користувачами. Цей аналіз надав мені цінні інсайди для розробки ефективної стратегії розвитку моєї онлайн платформи курсів та навчання, що сприятиме її успішному впровадженню та конкурентоспроможності на ринку.

Впровадив систему для платформи онлайн-навчання з використанням фреймворків Vue.js та Laravel. У процесі впровадження системи було реалізовано ключові функції, такі як реєстрація та авторизація користувачів, управління курсами, додавання нових курсів, їх перегляд та реєстрація на курси. Фронтенд частина була побудована на основі Vue.js, що забезпечило динамічний та інтерактивний інтерфейс користувача, зручний для навігації та використання. Бекенд частина на базі Laravel дозволила ефективно

організувати роботу з базою даних, забезпечити безпеку обробки даних та реалізувати бізнес-логіку додатку. Інтеграція обох фреймворків дозволила створити систему, яка відповідає сучасним вимогам до веб-розробки, демонструє високу продуктивність та надійність. Таким чином, впроваджена система підтвердила свою ефективність і доцільність для використання в контексті онлайн-навчання.

ВИСНОВКИ

Результатом роботи було створено веб-застосунок для навчання онлайн. Результати дослідження розробки та запуску моєї платформи онлайн курсів і навчання онлайн на основі Vue.js і Laravel свідчать про значний потенціал проекту та його можливості на ринку. Аналіз ринку показав стабільний ріст попиту на онлайн освітні послуги, що викликаний змінами у споживчих звичках та технологічними тенденціями. Це створює сприятливі умови для введення на ринок нової платформи для навчання.

Під час аналізу конкурентів були виявлені сильні та слабкі сторони провідних гравців у сфері онлайн освіти. Сильні сторони інших платформ включають широкий вибір курсів, високу якість навчального контенту та зручність використання, тоді як слабкі сторони пов'язані з високими вартостями та якістю обслуговування. Ці дані стали основою для визначення можливостей для покращення моєї платформи, включаючи розширення курсового контенту, підвищення якості навчання та оптимізацію вартості доступу до курсів.

Оцінка ефективності системи охоплювала аналіз продуктивності, надійності, зручності використання та безпеки. Високий рівень продуктивності та надійності досягався за рахунок оптимізації серверних ресурсів та інтерфейсу користувача. Зручність використання оцінювалася через тести юзабіліті та відгуки користувачів, що дозволяло постійно вдосконалювати платформу. Безпека системи гарантувалася за рахунок сучасних методів захисту даних та контролю доступу.

Отже, проведене дослідження підтверджує, що проект розробки та запуску моєї платформи онлайн курсів і навчання на основі Vue.js і Laravel має високий потенціал на ринку освітніх послуг.

Успіх розробки та впровадження моєї платформи онлайн курсів та навчання, що базується на Vue.js і Laravel, залежить від досягнення поставлених завдань. На першому етапі було важливо створити ефективну

архітектуру, що забезпечує високу швидкість та масштабованість. Vue.js і Laravel допомогли побудувати потужну систему з оптимізованою базою даних та стійким API.

Другим ключовим завданням стало створення зручного та естетичного інтерфейсу, який відповідає потребам користувачів. Було проведено тестування, щоб забезпечити максимальну зручність використання платформи.

Надійність і безпека були третім важливим аспектом. Використання Laravel дозволило впровадити механізми резервного копіювання та сучасні методи захисту даних, що забезпечило безпеку і захист інформації користувачів.

Це дозволило успішно впроваджувати платформу на конкурентному ринку онлайн освіти. Усі ці завдання були успішно виконані, що підтверджує ефективність стратегії розробки та високу якість реалізації моєї платформи онлайн курсів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документації Laravel. URL: https://laravel.com/docs/10.x (дата звернення: 10.05.2024)

2. Документація VueJS. URL: https://vuejs.org/guide/introduction.html (дата звернення: 11.05.2024)

3. Інформація про VUE. URL: https://medium.com/@jstify.community/vue-js-навіщо-він-нам-981b5e17af394. White, Emily. "Mastering Vue.js." Publisher, Year (дата звернення: 13.05.2024).

4. Переваги використання VUE. URL: https://foxminded.ua/vykorystannia-vue-v-laravel/ (дата звернення: 08.05.2024).

5. Переваги LARAVEL. URL: https://wezom.com.ua/ua/blog/17-preimuschestv-ispolzovanija-laravel-v-it-industrii (дата звернення: 08.05.2024).

6. Розробка з використанням Laravel. URL: https://promoter.net.ua/articles/rozrobka-veb-zastosunkiv-z-vikoristanniam-laravel-persii-krok.html(дата звернення: 19.05.2024).

7. Топ безкоштовних онлайн платформ. URL: https://www.ukrinform.ua/rubric-yakisne-zhyttia/3387961-7-bezkostovnih-nlajnplatform-dla-samoosviti.html (дата звернення: 21.05.2024).

8. Capterra. URL: https://www.capterra.com/learning-management-system-software/(дата звернення: 20.05.2024).

9. Courses online. URL: https://www.coursera.org/ (дата звернення: 01.06.2024).

10. Laravel для початківців. URL: https://foxminded.ua/laravel-dlia-pochatkivtsiv/(дата звернення: 02.05.2024).

11. Vue.js Documentation. URL: https://ua.vuejs.org/guide/introduction.html (дата звернення: 04.06.2024)