МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-07

Дипломний проєкт

здобувача освіти денної форми навчання

РП.07.03.000.ДП

ГАВРИЛЮКА ВЛАДИСЛАВА СЕРГІЙОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Інженерія програмного забезпечення»**

Група: **4РП-07**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка універсальної веб-орієнтованої платформи з онлайн курсів для навчання та перевірки знань

Проєктний матеріал складається з пояснювальної записки на **81** сторінках та графічного (презентаційного) матеріалу на **11** аркушах (слайдах).

Дипломник _____ (Гаврилюк В. С.)

Керівник _____ (Жадан А. С.)

Консультанти:

з економічного розділу _____ (Іванченков В. С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н. І.)

з нормоконтролю _____ (Петрашова В. І.)

старший консультант _____ (Кривченко Ю. В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю. В.)

Завідувач відділення _____ (Скорнякова О. В.)

Захист «27» 06 2024 р. Протокол ЕК № 3

Оцінка ЕК добре/85 б

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І. В.*

"15" 01 *2024 року*

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Гаврилюку Владиславу Сергійовичу*

1. Тема проєкту *Розробка універсальної веб-орієнтованої платформи з онлайн курсів для навчання та перевірки знань*

Затверджена наказом по коледжу від "02" *листопада* *2023 р.*, наказ № *244-А2-ОД*

2 Термін здачі закінченого проєкту *10 червня, 2024 р*

3. Вихідні дані до проєкту

1. Передбачити графічний дизайн користувача (GUI) засобами фреймворку Bootstrap

2. Передбачити дружній користувальницький досвід (UX)

3. Розробити Front-End частину веб-застосунку засобами Vue.js

4. Створити базу даних засобами СУБД SQLite

5. Розробити Back-End частину веб-застосунку засобами express.js

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

1. Аналіз предметної області. 2. Технології та засоби розробки (проєктування).

3. Проєктування веб-дизайну. 4. Проєктування архітектури веб-застосунку.

5. Розробка FullStack-застосунку. 6. Тестування створеного веб-застосунку.

7. Економічний розрахунок. 8. Аспекти охорона праці та техніки безпеки

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація Power Point – 11 слайдів

(Основні положення; Схема технологій; Схема UX-мапи; Схема БД;

Схема клієнт-серверної архітектури; Функціональність системи;

Процес розробки; Програмний код на отримання матеріалів курсу;

Скриншоти сторінок з курсами та матеріалами)

6. Консультанти по проєкту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Жадан А. С.*		
Економічний розділ	*Іванченков В. С.*		
Розділ охорони праці	*Чорновол Н. І.*		
Нормоконтроль	*Петрашова В. І.*		
Старший консультант	*Кривченко Ю. В.*		

7. Дата видачі завдання 15.01.2024

Керівник *Жадан А. С.* (підпис)

Завдання прийняв до виконання *Гаврилюк В. С.* (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту (роботи)	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Формування вступу*	*29.04.24*	виконано
2	*Аналіз предметної області*	*10.05.24*	виконано
3	*Підбір технічної літератури*	*19.05.24*	виконано
4	*Вибір технологій та засобів розробки (проєктування)*	*20.05.24*	виконано
5	*Проєктування веб-інтерфейсу веб-застосунку*	*22.05.24*	виконано
6	*Проєктування архітектури веб-застосунку*	*24.05.24*	виконано
7	*Розробка FullStack-застосунку*	*27.05.24*	виконано
8	*Тестування створеного веб-застосунку*	*29.05.24*	виконано
9	*Оформлення пояснювальної записки*	*31.05.24*	виконано
10	*Оформлення графічної (презентаційної) частини*	*01.06.24*	виконано
11	*Економічний розрахунок*	*02.06.24*	виконано
12	*Опис охорони праці та техніки безпеки*	*09.06.24*	виконано
13	*Аналіз результатів проєктування*	*13.06.24*	виконано
14	*Підготовка доповіді для захисту*	*16.06.24*	виконано

Дипломник (підпис)

Керівник (підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 07. 03 000. 00 ДП	Дипломний проєкт		
А4			РП 07. 03 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 07. 03 000. 00 ДП ПЗ		
Розроб.		Гаврилюк В.С.		10.06	Розробка універсальної веб-орієнтованої платформи з онлайн курсів для навчання та перевірки знань	Літ.	Аркуш 4 / Аркушів 81
Перевірив		Жадан А.С.		10.06.		Н Д П	
Н. Контр.		Петрашова В.І.		10.06	ВСП "ОТФК ОНТУ" гр.4РП-07		
Затверд.		Кривченко Ю.В.		10.06			

ЗМІСТ

ВСТУП

В епоху стрімкого розвитку цифрових технологій онлайн-навчання набуває все більшого значення. Постійне зростання попиту на дистанційні освітні програми стимулює створення ефективних і інноваційних рішень для організації освітнього процесу. Веб-платформи відіграють ключову роль у наданні зручних, швидких та ефективних інструментів для навчання та оцінювання знань.

Такі системи можуть включати різноманітні компоненти, зокрема інтерактивні курси, системи управління навчальними матеріалами, інструменти для тестування та аналізу успішності. Вони дозволяють учасникам освітнього процесу ефективно керувати своїм навчанням, підвищуючи продуктивність і забезпечуючи високий рівень обслуговування.

Використання автоматизованих веб-платформ дозволяє швидко адаптуватися до змін у навчальних програмах, оптимізувати процеси навчання та прискорювати оцінювання знань. Це підвищує якість освіти і конкурентоспроможність освітніх закладів у динамічному цифровому середовищі.

Платформи для дистанційного навчання можуть мати функції для перегляду та організації навчальних матеріалів, управління фінансами, звітності перед адміністрацією та аналізу освітніх тенденцій. Використання різних ролей у системі дозволяє ефективно розподіляти обов'язки між користувачами, надаючи доступ тільки до необхідної інформації та функцій для кожного.

Основна мета даної роботи полягає у створенні універсальної веб-платформи для онлайн-курсів, яка використовуватиме передові технології для оптимізації основних аспектів освітнього процесу. Система повинна забезпечувати зручні та ефективні інструменти для управління курсами, проведення оцінювання та аналізу даних, що дозволить учасникам навчального процесу підтримувати високий рівень сервісу та конкурентоспроможності.

Під час дослідження основних проблем предметної області, аналізу існуючих рішень та інструментів розробки було застосовано системний підхід. Для створення бази даних використовувалася технологія реляційних баз даних SQL. Для розробки програмного продукту застосовувалися технології об’єктно-орієнтованого програмування на мовах Node.js та JavaScript.

Практичне значення отриманих результатів полягає у тому, що розроблена платформа стане корисною для освітніх установ та студентів, допомагаючи їм ефективно організовувати освітній процес та забезпечувати високий рівень обслуговування.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз предметної області

Онлайн-платформа для навчання та перевірки знань “Udemy”.

“Udemy” – це популярна онлайн-платформа для навчання, що пропонує курси з широкого спектру тем, включаючи програмування, бізнес, маркетинг, дизайн, особистісний розвиток та багато іншого. Курси на Udemy створюються незалежними інструкторами, що дозволяє мати доступ до різноманітного контенту та навчальних матеріалів. Користувачі можуть обирати курси, які відповідають їхнім інтересам та професійним потребам, і проходити їх у зручний для себе час [1].

Основні характеристики “Udemy”:

Широкий вибір курсів: Платформа пропонує понад 155,000 курсів з різних тем.

Доступність: Користувачі можуть отримати доступ до курсів з будь-якої точки світу.

Гнучкість навчання: Курси доступні в режимі он-деманд, що дозволяє навчатися у зручний для себе час.

Різноманітність форматів: Курси можуть включати відео-лекції, статті, завдання та інтерактивні тести.

Сертифікація: Багато курсів пропонують сертифікати про завершення, які можуть бути корисними для професійного розвитку.

“Udemy” стає популярною серед тих, хто бажає отримати нові знання та навички, не виходячи з дому, та забезпечує зручний формат навчання для зайнятих професіоналів і студентів.

На рис. 1.1 відображено веб-сайт “Udemy”.

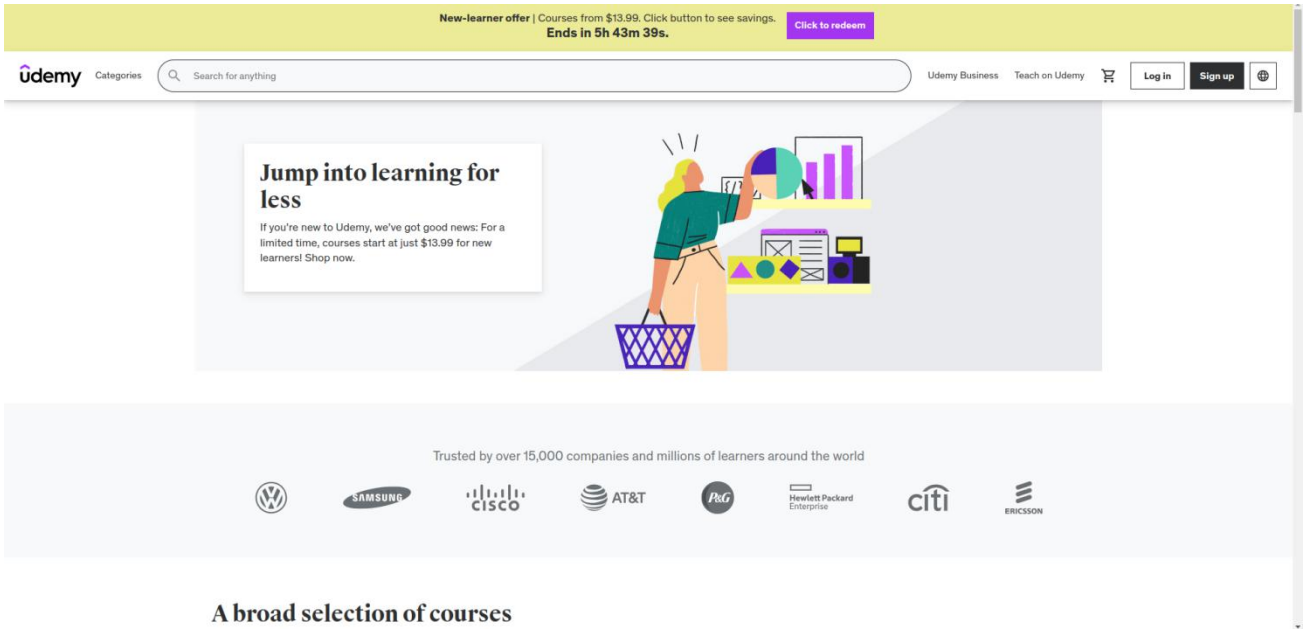


Рисунок 1.1. Веб-сайт “Udemy”

Онлайн-платформа для навчання “Coursera”.

“Coursera” – це відома онлайн-платформа для навчання, яка пропонує курси, спеціалізації, сертифікати та дипломні програми від провідних університетів та компаній з усього світу. Курси на Coursera розробляються у співпраці з університетами та компаніями, такими як “Stanford”, “Yale”, “Google”, “IBM” та іншими, що гарантує високий рівень якості навчальних матеріалів [2].

Основні характеристики Coursera:

Широкий вибір курсів: Платформа пропонує тисячі курсів з різних дисциплін, включаючи технології, бізнес, мистецтво, науку, особистісний розвиток та інші.

Партнерство з провідними установами: Курси розробляються у співпраці з відомими університетами та компаніями, що забезпечує актуальність та високий рівень навчання.

Гнучкість навчання: Курси доступні онлайн, що дозволяє навчатися у зручний для користувачів час і з будь-якого місця.

Сертифікація: Після успішного завершення курсу користувачі можуть отримати сертифікат, який можна додати до резюме або профілю LinkedIn для підтвердження нових навичок.

Можливість отримання дипломів: Coursera також пропонує онлайн-програми для отримання ступенів бакалавра та магістра в різних галузях.

Спеціалізації та професійні сертифікати: Курси об'єднуються в спеціалізації, що дозволяють користувачам глибше вивчати певну тему та отримувати професійні сертифікати.

“Coursera” є чудовим вибором для тих, хто прагне отримати нові знання та навички, підвищити свою кваліфікацію або навіть здобути нову професію завдяки онлайн-навчанню.

На рис. 1.2 відображено веб-сайт “Coursera”.

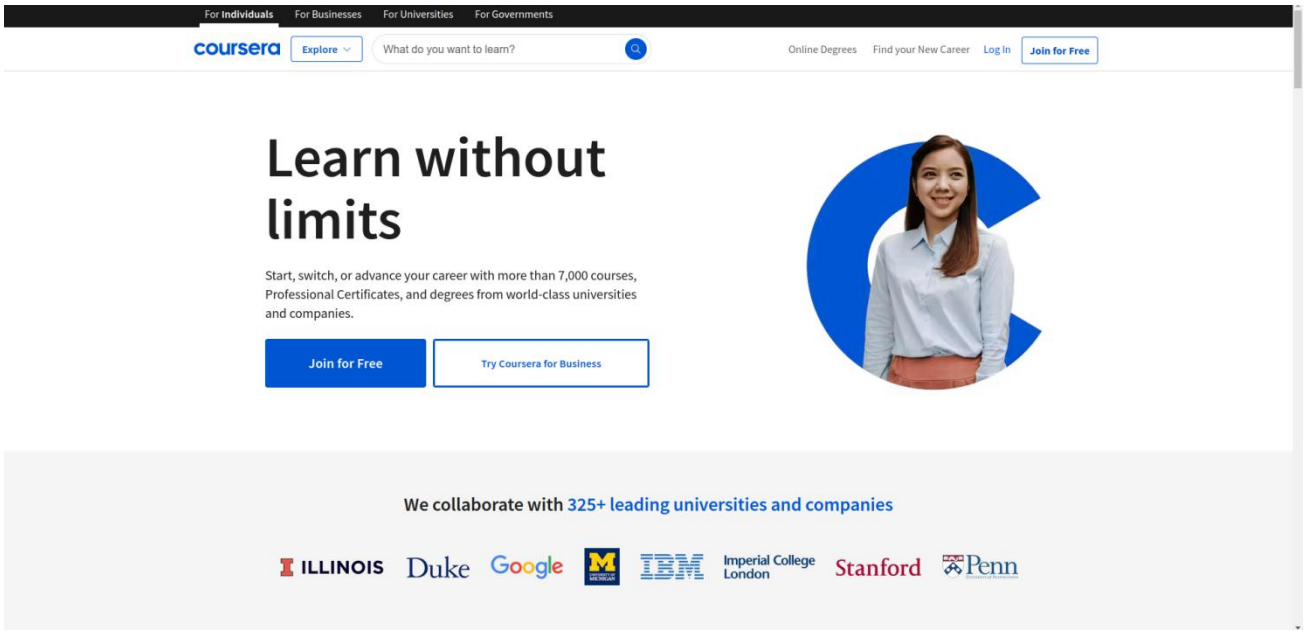


Рисунок 1.2. Веб-сайт “Coursera”

Онлайн-платформа для навчання “Prometheus”.

“Prometheus” – це онлайн-платформа, яка спеціалізується на наданні українських курсів та навчальних матеріалів. Ця платформа створена для того, щоб забезпечити доступ до якісної освіти українською мовою у відповідності з актуальними потребами ринку праці та сучасними технологіями [3].

Основні особливості платформи “Prometheus”:

Українськомовний контент: “Prometheus” пропонує широкий вибір українських курсів з різних галузей, що дозволяє користувачам здобувати знання рідною мовою.

Різноманіття предметів: На платформі доступні курси з різних предметів, включаючи технології, мистецтво, бізнес, мови, науку та багато іншого.

Ефективне навчання: Курси розробляються професійними викладачами та експертами відповідних галузей, що гарантує високий рівень якості навчання.

Гнучкий графік: Користувачі можуть навчатися у зручний для себе час та темп, забезпечуючи максимальний комфорт та ефективність навчання.

Сертифікація: Після успішного завершення курсів користувачі отримують сертифікати, які можна використовувати для підтвердження отриманих знань.

“Prometheus” надає можливість українським користувачам отримувати якісну освіту від відомих експертів та підвищувати свої навички у відповідності з потребами сучасного ринку праці.

На рис. 1.3 відображено веб-сайт “Prometheus”.

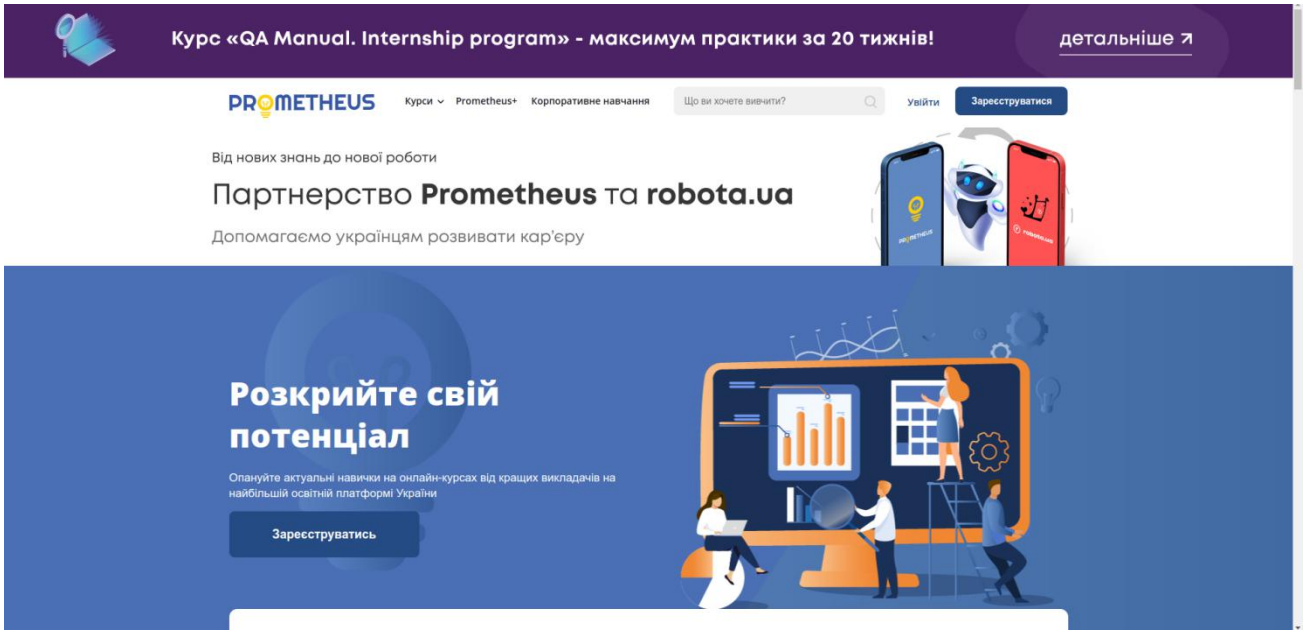


Рисунок 1.3. Веб-сайт “Prometheus”

У табл. 1.1. відображене порівняння платформ “Udemy”, “Coursera” та “Pro metheus”.

Таблиця 1.2. Порівняльна таблиця аналогів

Особливості	**“Udemy”**	**“Coursera”**	**“Prometheus”**
Мови	Багато мов, включаючи англійську та інші	Багато мов, включаючи англійську та інші	Українська
Вибір курсів	Понад 155,000 курсів з різних тем	Тисячі курсів та спеціалізацій	Широкий вибір українських курсів з різних тем
Партнери	Незалежні інструктори, компанії, університети	Університети, компанії та інші партнери	Українські компанії, університети та інші
Типи курсів	Різноманітні формати, від відео до текстових лекцій	Курси, спеціалізації, сертифікати, дипломні програми	Курси, можливість отримання сертифікатів
Гнучкість навчання	Навчання на вимогу, доступно 24/7	Гнучкість часу, можливість навчання на вимогу	Гнучкий графік навчання, доступність 24/7
Сертифікація	Сертифікати про завершення курсів	Сертифікати про завершення курсів та програм	Сертифікати про успішне завершення курсів

Висновок з порівняльної таблиці платформ Udemy, Coursera та Prometheus:

Усі три платформи – “Udemy”, “Coursera” та “Prometheus” є популярними онлайн-ресурсами для навчання, кожна з яких має свої унікальні особливості та переваги.

“Udemy” відома своїм широким вибором курсів і різноманітністю форматів навчання. Вона пропонує багато мов та дозволяє навчатися на вимогу, що дозволяє користувачам вибирати курси залежно від їхнього графіка та потреб.

“Coursera” співпрацює з провідними університетами та компаніями, що забезпечує високий рівень якості навчальних матеріалів. Крім того, на Coursera можна здобути сертифікати, спеціалізації та навіть дипломи в різних галузях.

“Prometheus”, з іншого боку, спеціалізується на україномовному контенті та пропонує широкий вибір українських курсів. Це може бути особливо корисно для тих, хто шукає освітній матеріал українською мовою.

Отже, вибір між цими платформами залежить від потреб та вподобань користувача. Якщо важлива гнучкість та широкий вибір, то Udemy може бути хорошим варіантом. Якщо потрібні вищі освітні програми та сертифікація, то Coursera може відповісти на ці потреби. А для тих, хто шукає україномовні курси, Prometeus буде чудовим вибором.

1.1.1 Технології та засоби розробки

Топ технологій технологій на думку Stack Oveflow у 2023 році:

Stack Overflow вважається авторитетним джерелом завдяки декільком ключовим факторам. По-перше, це велика та популярна платформа, яка збирає в собі велику кількість активних користувачів, включаючи професіоналів з різних галузей програмування. По-друге, головною метою Stack Overflow є сприяння обміну знаннями та досвідом між програмістами, що стимулює високий рівень експертизи серед учасників. Крім того, платформа забезпечує можливість швидкого та зручного пошуку вже існуючих відповідей на питання, що раніше були задані іншими користувачами, що сприяє швидкому розв'язанню проблем. Крім того, активна спільнота користувачів допомагає забезпечити актуальні та надійні відповіді на різноманітні питання з програмування та розробки програмного забезпечення. У кінцевому підсумку, завдяки цим чинникам, Stack

Overflow є довіреним джерелом інформації та експертної допомоги для програмістів у всьому світі [4].

На рис. 1.4 відображено топ технологій технологій.

Рисунок 1.4. Топ технологій

За даними Stack Oveflow, Node.js та React.js вважаються найпоширенішими веб-технологіями серед розробників. Професійні програмісти активно використовують обидві, а ті, хто вивчають програмування, надають перевагу Node.js більш, ніж React (52% проти 48%). На наступних позиціях за популярністю знаходяться jQuery та Express, причому jQuery використовується більше професійними розробниками, ніж тими, хто вивчає програмування (24% проти 18%), у той час як Express використовується більше тими, хто вивчає, ніж професіоналами (25% проти 20%). Загалом, ці дані свідчать про те, що Node.js та React.js залишаються ключовими технологіями для веб-розробки, проте рівень їх використання може залежати від досвіду програмістів.

Стек обраних технологій:

SQLite: Легкий та вбудований SQL двигун баз даних, який надає простий у використанні, однофайловий механізм зберігання даних. Він дозволяє створювати, читати, оновлювати та видаляти дані без необхідності встановлення окремого сервера баз даних [5].

Express.js: Ппопулярний веб-фреймворк для Node.js, який дозволяє швидко створювати серверні застосунки та API. Він має мінімалістичний та гнучкий дизайн, що дозволяє розробникам ефективно обробляти маршрути, обробляти запити та відповіді, керувати статусами та багато іншого [6].

Vue.js: Прогресивний JavaScript фреймворк для розробки користувацьких інтерфейсів та односторінкових додатків. Він відомий своєю простотою та ефективністю, дозволяючи легко створювати веб-інтерфейси за допомогою декларативного синтаксису та компонентної архітектури [7].

Bootstrap: Потужна та популярна HTML, CSS та JavaScript бібліотека, яка містить набір готових компонентів та інструментів для швидкого розроблення сучасних та адаптивних веб-інтерфейсів. Вона дозволяє зосередитися на дизайні та структурі сторінок, забезпечуючи при цьому високу якість та сумісність з різними браузерами [8].

На рис. 1.5 відображено стек технологій. Схема створена засобами бзекоштовного сервісу draw.io.

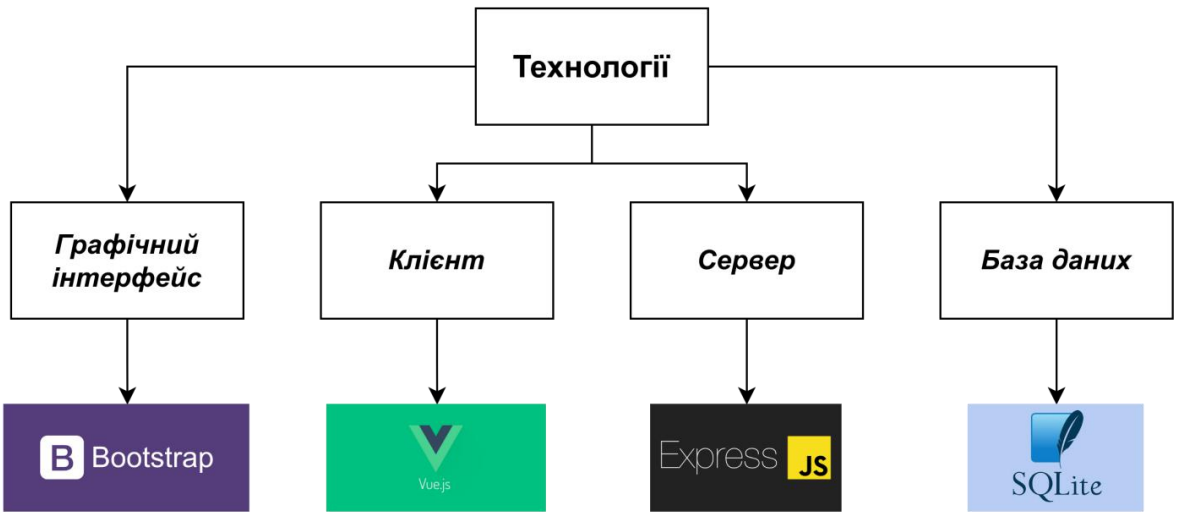


Рисунок 1.5. Стек технологій

Основні особливості Node.js:

Асинхронність: В основі Node.js лежить асинхронне програмування, що дозволяє ефективно обробляти багато запитів без блокування виконання інших операцій.

Платформонезалежність: Node.js підтримується на багатьох платформах, включаючи Windows, macOS і Linux.

Висока швидкість: Завдяки використанню двигуна V8 від Google, Node.js забезпечує високу швидкість виконання коду.

Розширюваність: Node.js має велику кількість модулів у вигляді пакетів npm, що дозволяє розширити його функціональність і використовувати готові рішення для розробки.

Спільнота розробників: Node.js має активну спільноту розробників, яка постійно вносить удосконалення та нові функції у платформу.

Node.js використовується для розробки різноманітних застосунків, таких як веб-сервери, мережеві застосунки, API, мікросервіси та інші, і знаходить широке застосування у розробці сучасних веб-проєктів.

У табл. 1.2 відображене порівняння мов програмування.

Таблиця 1.2. Порівняльна характеристика мов програмування

Особливості	**Node.js**	**Python**
Мова	JavaScript	Python
Використання	Серверна розробка, веб-застосунки, мережеві застосунки	Веб-розробка, аналіз даних, наукові обчислення, штучний інтелект
Асинхронність	Основна особливість, побудована на асинхронному програмуванні	Можливе асинхронне програмування, але не є основним

Продовження таблиці 1.2

Синтаксис	Заснований на JavaScript	Простий та зрозумілий синтаксис
Веб-фреймворки	Express.js, Koa.js, Hapi.js та інші	Django, Flask, Pyramid та інші
Швидкодія	Висока швидкодія завдяки асинхронності	Відмінна швидкодія, особливо у деяких областях, таких як обробка тексту та аналіз даних
Розширюваність	Велика кількість пакетів у вигляді модулів npm	Велика кількість бібліотек та модулів у вигляді пакетів pip
Використання відомих компаній	Netflix, PayPal, LinkedIn та інші	Google, Instagram, Spotify та інші

Обрані технології мають великий потенціал для розробки онлайн платформи для невеликих курсів та перевірки знань. SQLite є ідеальним вибором для зберігання даних, оскільки він легкий, вбудований і не потребує окремого сервера баз даних, що робить його зручним для маленьких проєктів. Express.js, як популярний веб-фреймворк для Node.js, дозволяє швидко створювати серверні застосунки та API, що є важливим аспектом для реалізації функціоналу платформи. Vue.js, як прогресивний фреймворк для розробки інтерфейсів, забезпечить користувачам зручний та ефективний досвід взаємодії з платформою, завдяки своїй простоті та компонентній архітектурі. Bootstrap, як потужна бібліотека для розробки адаптивних веб-інтерфейсів, допоможе створити стильний та професійний дизайн для платформи, забезпечуючи високу якість та сумісність з різними пристроями та браузерами. Загалом, ці технології відмінно поєднуються для створення функціональної та естетичної платформи для навчання та тестування знань.

1.2 Проєктування навчальної платформи

1.2.1 Технічне завдання на розробку

Аспекти технічного завдання:

Мінімалістичний графічний інтерфейс користувача: Платформа має простий та зрозумілий дизайн, без зайвих деталей та елементів, що забезпечує зручність та легкість використання для користувачів.

Мінімалістичний користувальницький досвід: Користувачі можуть легко навігуватися по платформі та виконувати необхідні дії без зайвих складнощів чи заплутаності.

Наявність теоретичного матеріалу до курсів: Кожен курс має відповідний теоретичний матеріал, який допомагає користувачам засвоювати теоретичні основи перед переходом до практичних завдань.

Наявність тестів до курсів: Кожен курс включає набір тестів або вправ для перевірки знань після закінчення теоретичного матеріалу або після кожного розділу.

Наявність реєстрації та авторизації: Користувачі можуть створити обліковий запис на платформі шляхом реєстрації та потім увійти до свого облікового запису за допомогою авторизації.

Наявність кабінету користувача: Кожен користувач має особистий кабінет, де він може переглядати свій прогрес у курсах, результати тестів, а також редагувати особисту інформацію.

Наявність кабінету адміністратора: Адміністратор має доступ до спеціального кабінету з розширеними можливостями управління платформою, включаючи керування користувачами, курсами, завданнями та іншими аспектами.

Видача сертифікату: Після успішного завершення курсу користувач отримує сертифікат, який підтверджує його знання та досягнення.

Підтримка української мови: Платформа має можливість використання української мови інтерфейсу для зручності користувачів з україномовним бекграундом.

Запланована навігація у веб-системі.

На рис. 1.6 відображено UX навігацію. Схема створена засобами бзекоштовного сервісу draw.io.

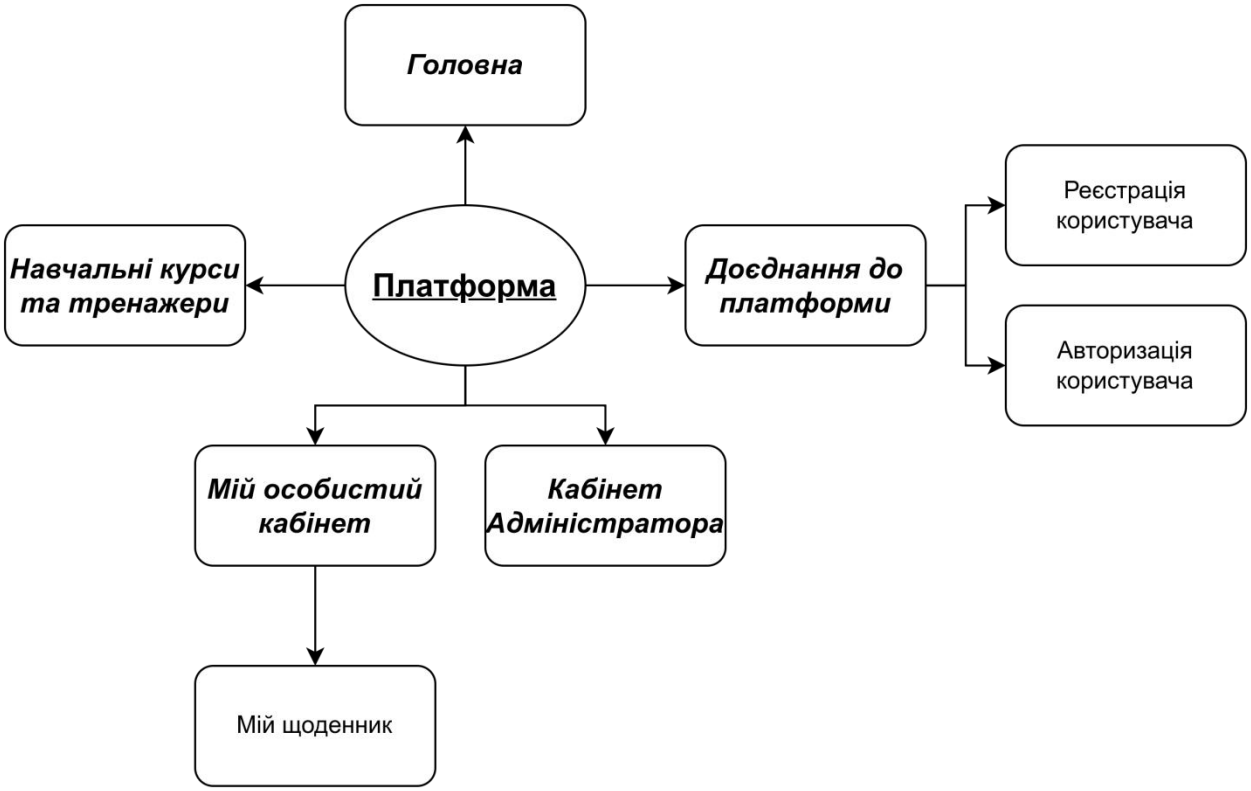


Рисунок 1.6. UX Навігація

1.2.2 Проєктування дизайну

Загальний шаблон веб-сторінок у веб-системі:

Навбар: Висока частина веб-сторінки, яка містить навігаційні елементи, такі як посилання на інші сторінки, логотипи, меню та інші корисні елементи для навігації по сайту. Навбар зазвичай розташовується у верхній частині сторінки і може залишатися видимим на всій довжині сторінки або звертатися у вертикальне меню на більш маленьких екранах [9].

Хеадер: Верхня частина веб-сторінки, яка може включати в себе не лише навбар, але й додаткові елементи, такі як заголовок сторінки, зображення або відео, важливі повідомлення чи кнопки дій. Хеадер зазвичай є першим, що бачить користувач при відкритті сторінки і встановлює загальний тон та стиль [9].

Контент: Основна частина веб-сторінки, яка містить інформацію, яку користувач шукає або очікує знайти. Сюди входять текст, зображення, відео, таблиці, графіки та інші елементи, які передають контент або функціональність сторінки.

Футер: Це нижня частина веб-сторінки, яка містить додаткову інформацію або елементи навігації, такі як посилання на інші сторінки, контактні дані, соціальні медіа, копірайт, а також інші важливі відомості. Футер зазвичай міститься внизу сторінки і може бути фіксованим (залишатися видимим при прокручуванні) або динамічним (з'являтися тільки після досягнення кінця сторінки) [9].

На рис. 1.7 відображено дизайн десктопного навбару

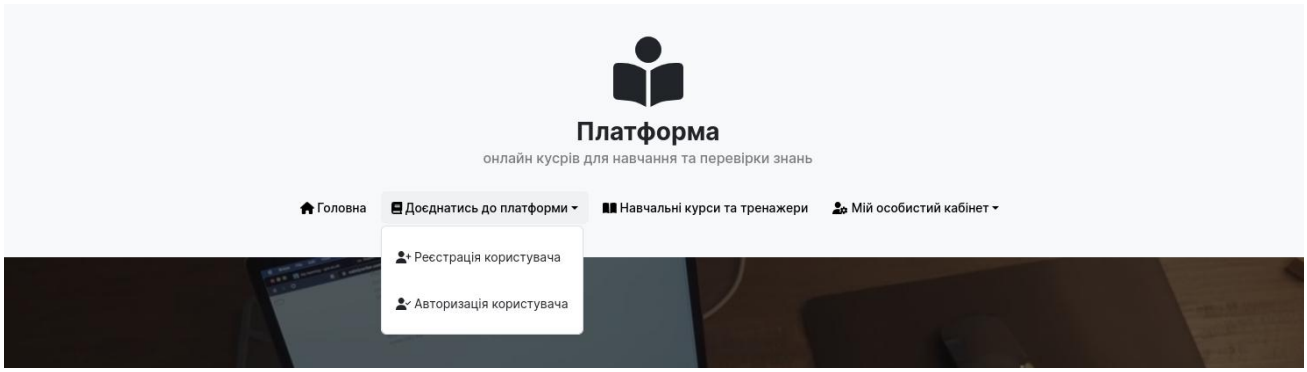


Рисунок 1.7. Дизайн десктопного навбару

На рис. 1.8 відображено дизайн мобільного навбару.

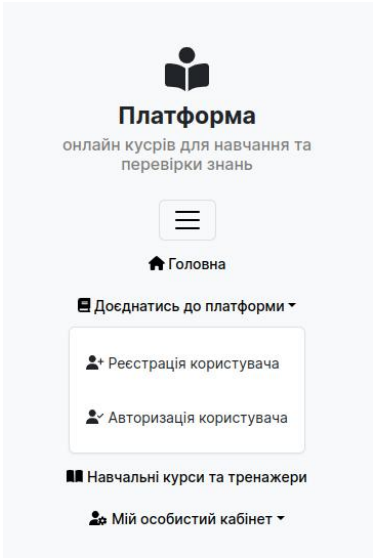


Рисунок 1.8. Дизайн мобільногонавбару

На рис. 1.9 відображено дизайн хеадеру.

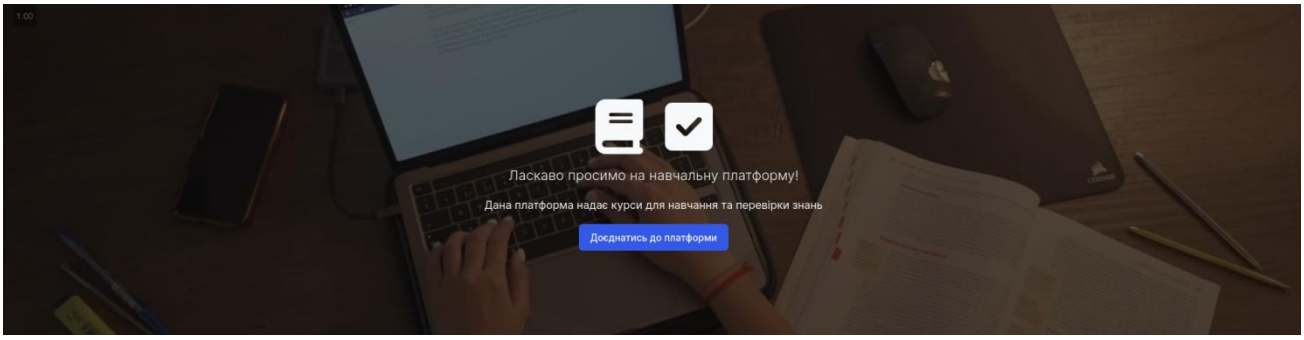


Рисунок 1.8. Дизайн хеадеру

На рис. 1.10 відображено дизайн футеру.

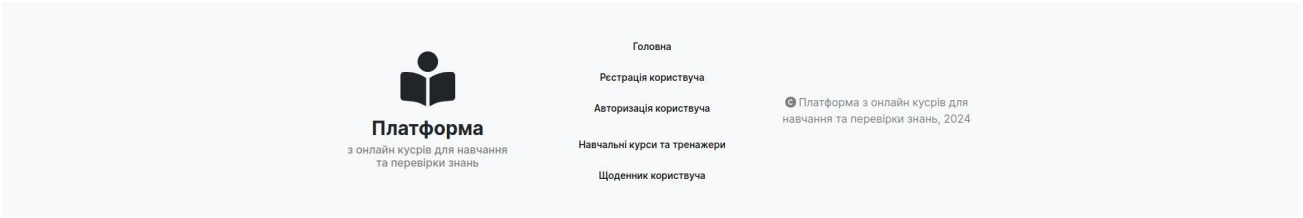


Рисунок 1.8. Дизайн футеру

1.2.3 Проєктування бази даних веб-системи

На рис. 1.11 показана схема бази даних веб-системи. Схему бази даних створено в умовнобезкоштовному менеджері баз даних dbeaver.

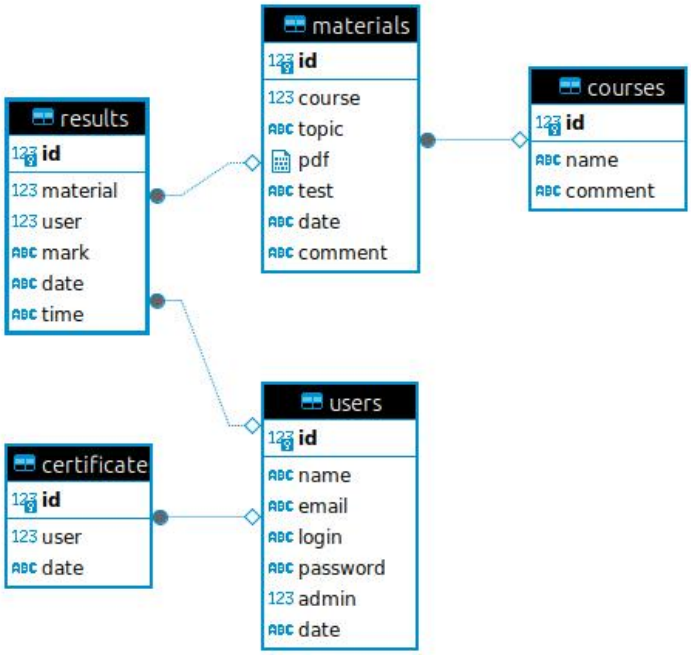


Рисунок 1.11. Схема бази даних

Таблиці і поля бази даних:

Таблиця «users» зберігає користувачів:

Id – ідентифікатор користувача;

Name – ПІБ;

Email – електрона пошта;

Login – логін;

Password – хешований пароль;

Admin – перевірка на права адміністратора;

Date – дата реєстрації користувача;

Таблиця «courses» зберігає інформацію про курси за мовами програмування:

Id – ідентифікатор курсу;

Name – назва курсу;

Таблиця «materials» зберігає матеріали курсів:

Id – ідентифікатор навчального матаріалу;

Course – посилання на курс;

Topic – тема навчального матеріалу;

Comment – опис навчального матеріалу;

Pdf – текст навчального матеріалу у форматі pdf;

Topic – тема навчального матеріалу;

Test – тест до навчального матеріалу у текстовому форматі;

Date – дата створення навчального матеріалу.

Таблиця «results» зберігає результати користувачів по навчальним матеріалам:

Id – ідентифікатор результату;

Material – посилання на навчальний матеріал;

User – посилання на користувача;

Mark – оцінка за тест;

Time – затрачений час на проходження тесту;

Date – дата проходження тесту.

Таблиця «certificats» зберігає сертифікати користувачів у разі проходження всіх матеріалів курсів:

Id – ідентифікатор сертифікату;

User – посилання на користувача;

Date – дата отримання сертифікату.

На рис. 1.12 відобрадено перелік створених таблиць. Скриншот створено в умовнобезкоштовному менеджері баз даних dbeaver.

Рисунок 1.12. Перелік таблиць

1.2.4 Проєктування архітектури веб-системи

У поточному проєкті використовується безкоштовна реляційна однофайлова база даних SQLite разом з безкоштовним універсальним графічним

редактором баз даних DBeaver Community. На серверній стороні проєкту маємо RESTful API, яке є шаром між базою даних і клієнтським застосунком. Цей інтерфейс реалізується через протокол верхнього рівня HTTP і спирається на фреймворк Express.js. Використовується високорівнева мова програмування NodeJS. Авторизація в застосунку здійснюється за допомогою JWT токенів, що передаються у заголовках HTTP протоколу.

На клієнтській стороні проєкт представлений веб-застосунку, побудованим на прогресивному (реактивному) фреймворку Vue.js. Цей фреймворк використовує JavaScript, а його реактивність означає асинхронний підхід до отримання даних з API, не вимагаючи повторного завантаження сторінки. Графічний інтерфейс веб-застосунку, що генерується за допомогою адаптивного фреймворку Bootstrap, автоматично адаптується під різні розміри екрану, включаючи мобільні, планшетні та десктопні версії.

Цей підхід є наразі найактуальнішим, оскільки в минулому серверний застосунок повертав готові HTML сторінки з даними, а не JSON файли [10].

На рис. 1.13 відображенко архітектуру Full-Stack проєкту. Схема створена засобами бзекоштовного сервісу draw.io.

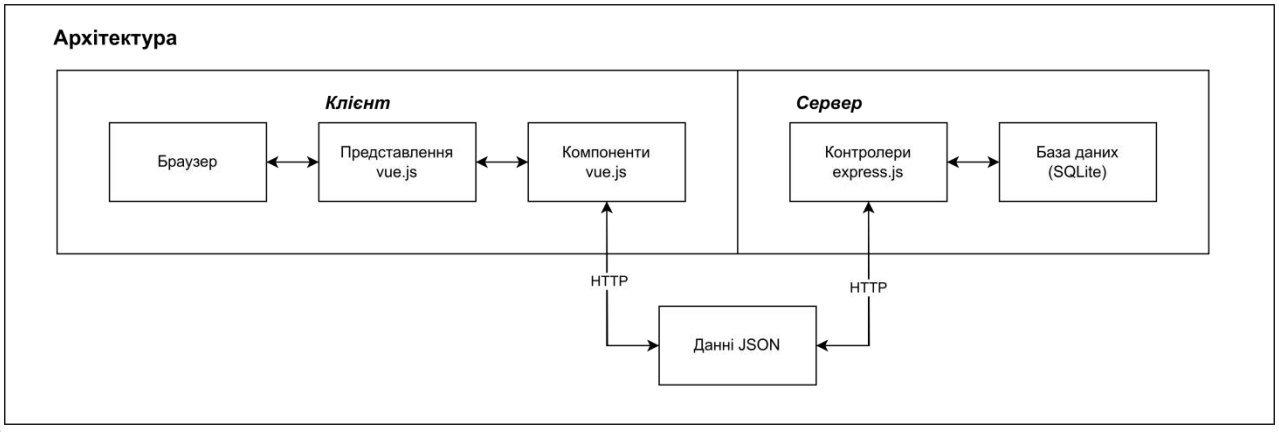


Рисунок 1.13. Загальна архітектура веб-системи

проєкт має наступну файлову структуру.

На рис. 1.14 відображено файлову структуру Full-Stack проєкту. Скриншот зроблений в безкоштовному редакторі коду Microsoft Visual Studio Code.

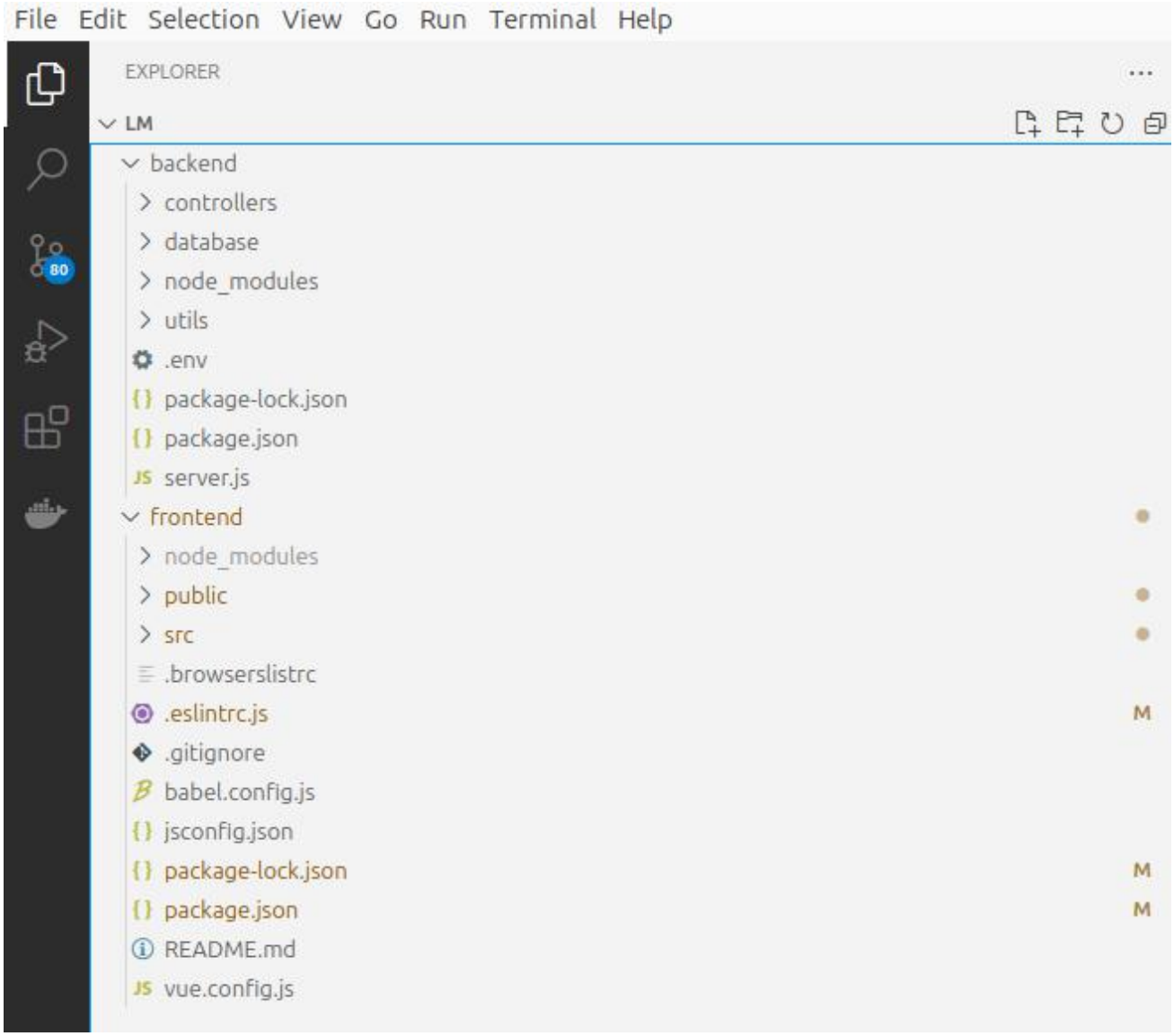


Рисунок 1.14. Файлова архітектура мікро-фреймворку PHP

Опис файлову структури проєкту.

"./frontend": Тут міститься фронтенд частина проєкту, побудована на фреймворку Vue.js.

"./frontend/src": У цій текі знаходиться вихідний код фронтенду, включаючи компоненти, маршрути та інші складові.

"./frontend/node_modules": Це тека, де розміщені залежності, необхідні для роботи фронтенду, які встановлені за допомогою npm або yarn.

"./frontend/.git": Тут містяться дані системи керування версіями Git, що стосуються фронтенду проєкту.

"./frontend/public": У цій теці розташовані статичні ресурси, такі як зображення, шрифти та інші файлові активи, доступні для відображення клієнтам.

"./backend": Це тека, де знаходиться бекенд частина проєкту, побудована на фреймворку Express.js.

"./backend/database": Тут можуть знаходитися файли бази даних або конфігураційні файли, що стосуються зберігання даних.

"./backend/node_modules": Тут розміщені залежності, необхідні для роботи серверу, які встановлені за допомогою npm або yarn.

"./backend/utils": Це тека, де можуть знаходитися допоміжні функції або утиліти, які використовуються у серверній частині проєкту.

"./backend/controllers": Тут розташовані контролери, які обробляють запити від клієнтів та забезпечують відповідну відповідь.

1.3 Реалізація Full-Stack застосунку

Full-Stack переставляє собою програмне забезпечення, що охоплює як фронтенд (інтерфейс користувача), так і бекенд (серверну частину) частини розробки. Фронтенд відповідає за взаємодію користувача з програмою через браузер, використовуючи HTML, CSS та JavaScript, а бекенд обробляє запити, управляє базами даних і виконує бізнес-логіку на сервері, використовуючи мови програмування, як-от Node.js, Python або Ruby. Таким чином, full stack розробник володіє навичками для створення комплексних веб-застосунків, що включають обидві частини.

1.3.1 Створення бази даних

База даних SQLite.

Створимо базу даних за допомогою командного рядку.

На рис. 1.15 відображенко створену базу даних SQLite. Скриншот зроблений в безкоштовному редакторі коду Microsoft Visual Studio Code.

Рисунок 1.15. Створена база даних SQLite

1.3.2 Розробка серверної частини (Back-End API)

Серевер API у express.js.

Сервер представляє собою комп'ютер або програмне забезпечення, що обробляє запити від клієнтів і надає їм доступ до ресурсів або послуг, а API представляє собою набір правил і протоколів, що визначають способи взаємодії між різними програмами або сервісами. Вони пов'язані між собою таким чином, що сервер може надавати функції або дані через веб-інтерфейс, який використовує API для взаємодії з іншими програмами або клієнтами через мережу.

Нижче наведено код налаштування серверу у файлі backend/server.js.

```
// Create express app
var express = require("express")
var app = express()
var cors = require('cors')

const dotenv = require('dotenv');
dotenv.config();

// Server port
var HTTP_PORT = 8000
// Start server
app.listen(HTTP_PORT, () => {
    console.log("Server running on port %PORT%".replace("%PORT%",HTTP_PORT))
});

app.use(
    cors({origin: ['http://localhost:8080', 'http://127.0.0.1:8080']})
  );
```

```
// Root endpoint
app.get("/", (req, res, next) => {
    res.json({"message":"Ok"})
});

// Insert here other API endpoints
app.use(require('./controllers/users'));
app.use(require('./controllers/courses'));
app.use(require('./controllers/materials'));
app.use(require('./controllers/results'));
app.use(require('./controllers/certificates'));

// Default response for any other request
app.use(function(req, res){
    res.status(404);
});
```

Цей код створює сервер за допомогою Express, підключає модуль CORS для обробки CORS-запитів, зчитує конфігураційні дані з файлу .env, задає порт HTTP сервера (8000) і запускає його, виводячи повідомлення про запуск. Далі встановлюються обробники маршрутів: кореневий маршрут повертає JSON з повідомленням "Ok", а також підключаються інші API маршрути через відповідні контролери. Якщо запит не відповідає жодному маршруту, відправляється відповідь зі статусом 404.

Контролери API у express.js.

Контролери представляють собою частину програмного забезпечення, яка відповідає за обробку запитів від клієнтів і визначення логіки обробки цих запитів. API, з свого боку, є набором правил і протоколів, які визначають, як програми або сервіси можуть взаємодіяти один з одним. Вони пов'язані між собою таким чином, що контролери реалізують функції або логіку доступу до даних через веб-інтерфейс, що використовується API для взаємодії з іншими програмами або клієнтами через мережу.

На рис. 1.16 відображенко створені контролери. Скриншот зроблений в безкоштовному редакторі коду Microsoft Visual Studio Code.

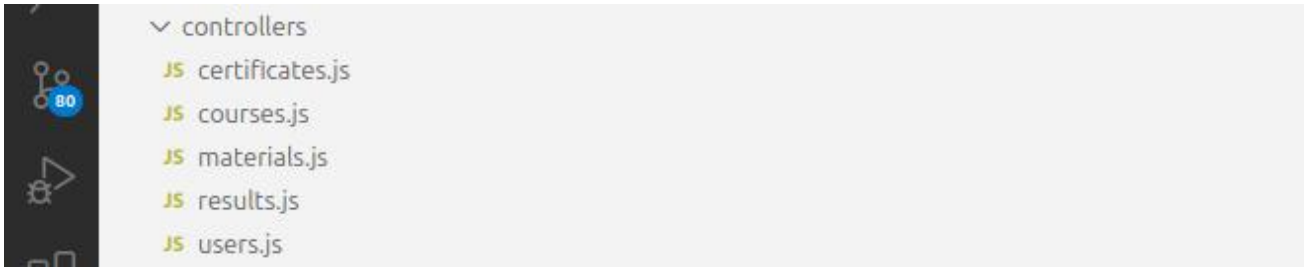


Рисунок 1.16. Контролери

Нижче наведено код контролеру матеріалів курсів у файлі backend/controllers/materials.js.

```
// Create router
var express = require('express')
var router = express.Router()

// Include DB
var db = require("../utils/db")

// Include Auth
var auth = require("../utils/auth")

// Include Test logic
var test = require("../utils/test")

// Middleware for parsing data
var bodyParser = require("body-parser")
router.use(bodyParser.urlencoded({ extended: false }))
router.use(bodyParser.json())

var fileUpload = require('express-fileupload');
router.use(fileUpload());

// Get a List of Materials by Course
router.get("/api/materials/:id", (req, res, next) => {
    if (auth.isUser(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var sql = `SELECT * FROM materials WHERE course = ?`
    var params = [req.params.id]

    db.all(sql, params, (err, rows) => {
        if (err) {
            res.status(400).json({ "error": err.message })
            return
        }

        res.json({
            "message": "Операція успішна",
            "data": rows
        })
    })
})

// Add an Material
router.post("/api/materials", (req, res, next) => {
```

```
            if (auth.isAdmin(req.headers.authorization) == false) {
                return res.status(401).json({ message: "Не авторизовано" })
            }

            var data = {
                course: req.body.course,
                topic: req.body.topic,
                comment: req.body.comment,
                pdf: req.files?.pdf?.data || null,
                test: req.body.test,
                date: new Date().toISOString().replace('T', ' ').replace('Z', '')
            }
            var sql = "INSERT INTO materials (course, topic, comment, pdf, test, date)
VALUES (?,?,?,?,?,?)"
            var params = [data.course, data.topic, data.comment, data.pdf, data.test,
data.date]

            db.run(sql, params, function (err, result) {
                if (err) {
                    res.status(400).json({"error": err.message})
                    return
                }
                res.json({
                    "message": "Операція успішна",
                    "data": data,
                    "id" : this.lastID
                })
            })
        })

        // Edit an Material by Id
        router.patch("/api/materials/:id", (req, res, next) => {
            if (auth.isAdmin(req.headers.authorization) == false) {
                return res.status(401).json({ message: "Не авторизовано" })
            }

            var data = {
                course: req.body.course,
                topic: req.body.topic,
                comment: req.body.comment,
                pdf: req.files?.pdf?.data || null,
                test: req.body.test,
                date: new Date().toISOString().replace('T', ' ').replace('Z', '')
            }
            var sql = `UPDATE materials set
            course = COALESCE(?,course),
            topic = COALESCE(?,topic),
            comment = COALESCE(?,comment),
            pdf = COALESCE(?,pdf),
            test = COALESCE(?,test),
            date = COALESCE(?,date)
            WHERE id = ?`
            var params = [data.course, data.topic, data.comment, data.pdf, data.test,
data.date, req.params.id]

            db.run(sql, params,
                function (err, result) {
                    if (err){
                        res.status(400).json({"error": res.message})
                        return
                    }
```

```
                res.json({
                    message: "Операція успішна",
                    data: data,
                    changes: this.changes
                })
    })
})

// Delete an Material by id
router.delete("/api/materials/:id", (req, res, next) => {
    if (auth.isAdmin(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var sql = "DELETE FROM materials WHERE id = ?"
    var params = [req.params.id]

    db.run(sql, params, function (err, Certificate) {
        if (err) {
            res.status(400).json({"error": res.message})
            return
        }
        res.json({"message":"Операція успішна", changes: this.changes})
    })
})

module.exports = router
```

Цей код визначає роутер для API, використовуючи Express.js, з налаштуванням middleware для обробки даних та завантажених файлів. Він містить обробники для отримання списку матеріалів за курсом, додавання нового матеріалу, оновлення та видалення матеріалу за його ідентифікатором. Кожен обробник перевіряє автентифікацію користувача перед здійсненням операції. У разі успішного виконання операцій, сервер повертає відповідь з повідомленням "Операція успішна" та відповідними даними. В разі помилки - відповідний статус та повідомлення про помилку. На кінці файлу роутер експортується для використання в основному застосунку.

Утилити API у express.js.

Створимо модуль авторизації для API.

Модуль авторизації відповідає за перевірку та обробку запитів авторизації від клієнтів, а також визначення логіки доступу до ресурсів на основі JWT токенів. JWT (JSON Web Token) представляє собою стандарт токенізації, що дозволяє безпечно передавати між сторонами інформацію у вигляді JSON об'єктів.

Нижче наведено код, який реалізує JWT авторизацію у файлі backend/utils/auth.js.

```
// JWT
var jwt = require('jsonwebtoken')

function isUser(header) {
    var token = header.split(' ')[1]

    if (!token) {
        return false
    }

    var decoded = jwt.verify(token, process.env.JWT_SECRET)

    var isUser = decoded.isUser

    if (isUser) {
        return true
    }
}

function getUserId(header)
{
    var token = header.split(' ')[1]

    if (!token) {
        return false
    }

    var decoded = jwt.verify(token, process.env.JWT_SECRET)

    var userId = decoded.userId

    if (userId) {
        return userId
    }
}

function isAdmin(header) {
    var token = header.split(' ')[1]

    if (!token) {
        return false
    }

    var decoded = jwt.verify(token, process.env.JWT_SECRET)

    var isAdmin = decoded.isAdmin

    if (isAdmin == 0)
    {
        return false
    }
    else
    {
        return true;
    }
}
```

```
module.exports = { isUser, getUserId, isAdmin }
```

Даний код модуля JWT авторизації використовує бібліотеку jwt для перевірки та розшифрування JWT токенів. Функції isUser, getUserId та isAdmin призначені для перевірки різних типів користувачів на основі JWT токенів. Через розбиття заголовка на токен та розшифрування його вмісту з використанням секретного ключа JWT, код перевіряє, чи існує користувач з певними правами доступу: користувач, адміністратор чи користувач з певним ідентифікатором. Результатом виконання функцій є логічне значення true чи false, в залежності від наявності та прав доступу користувача, або ідентифікатор користувача. Функції isUser та isAdmin повертають true для відповідних користувачів, в той час як getUserId повертає ідентифікатор користувача, або false у випадку невалідного токену чи відсутності правильних даних. Весь модуль експортується як об'єкт, що містить ці функції для використання в інших частинах програмного забезпечення.

1.3.3 Розробка клієнтської частини (Front-End App)

Конфігурація у vue.js.

Vue.js представляє собою JavaScript фреймворк для побудови користувацьких інтерфейсів, який дозволяє створювати веб-застосунки та односторінкові застосунки з інтерактивними інтерфейсами. Файл main.js відповідає за ініціалізацію та конфігурацію Vue.js застосунку, включаючи підключення компонентів, маршрутизацію, управління станом застосунку та інше.

Нижче наведено код налаштування основного файлу застосунку у файлі frontend/src/main.js.

```
import { createApp } from 'vue'
import App from './App.vue'
import './registerServiceWorker'
import router from './router'
import store from './store'

import apiClient from './plugins/api.js';

import 'bootstrap/dist/js/bootstrap.min.js';
import 'bootstrap/dist/css/bootstrap.min.css';
```

```
import 'bootswatch/dist/zephyr/bootstrap.min.css';

import '@/assets/css/styles.css';

import '@fortawesome/fontawesome-free/js/all.js';

const app = createApp(App);

app.use(store);
app.use(router);
app.mount('#app');

app.config.globalProperties.$api = apiClient;
```

Даний файл main.js відповідає за конфігурацію та ініціалізацію Vue.js застосунку. Він імпортує необхідні залежності, такі як основний компонент застосунку, сервіс роутингу, зберігання та плагіни API. Після цього файл встановлює залежності за допомогою методів use, які додають роутер та зберігання до застосунку. Далі він монтує головний компонент застосунку до DOM за допомогою методу mount. На останок, файл налаштовує глобальний об'єкт $api, який представляє сервіс для взаємодії з API, і додає його до глобальних властивостей застосунку.

Маршрутизація (роутинг) vue.js.

Маршрутизатор в Vue.js відповідає за визначення шляхів та перехоплення навігаційних дій користувача, включаючи відображення відповідних компонентів для кожного шляху. В основному файлі застосунку (main.js) ми встановлюємо маршрутизатор і підключаємо його до застосунку.

Нижче наведено код налаштування маршрутизатора у файлі router/ router.js.

```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'

const routes = [
{
path: '/',
name: 'home',
component: HomeView
},
{
path: '/signin',
name: 'signin',
component: () => import('../views/SignInView.vue')
},
{
path: '/signup',
```

```
name: 'signup',
component: () => import('../views/SignUpView.vue')
},
{
path: '/cabinet',
name: 'cabinet',
component: () => import('../views/CabinetView.vue')
},
{
path: '/panel',
name: 'panel',
component: () => import('../views/PanelView.vue')
},
{
path: "/courses",
name: "courses",
component: () => import('../views/CoursesView.vue')
},
{
path: "/materials/:id",
name: "materials",
params: true,
component: () => import('../views/MaterialsView.vue')
},
{
path: "/test/:id",
name: "test",
params: true,
component: () => import('../views/TestView.vue')
},
{
path: "/certificate/:id",
name: "certificate",
params: true,
component: () => import('../views/CertificateView.vue')
}
]

const router = createRouter({
history: createWebHistory(process.env.BASE_URL),
routes
})

router.beforeEach((to, from, next) => {
window.scrollTo(0, 0)
next()
})

export default router
```

У файлі router.js встановлюється маршрутизатор Vue.js за допомогою функції createRouter та створенням історії браузера з використанням createWebHistory. Визначаються шляхи та відповідні компоненти для них, такі як головна сторінка, сторінки входу та реєстрації, кабінет користувача, панель управління, перегляд курсів, матеріалів, тестів та сертифікатів. Інші шляхи вказують на відповідні компоненти для відображення вмісту. Кожен маршрут

має відповідне ім'я та шлях, які використовуються для навігації. Перед переходом до кожного маршруту виконується функція beforeEach, яка прокручує сторінку до верху. Остаточно маршрутизатор експортується для використання в головному файлі застосунку.

Інтерсептор vue.js.

Інтерсептор в Vue.js відповідає за перехоплення та обробку HTTP-запитів до сервера та відповідей від нього. У головному файлі застосунку (main.js) ми підключаємо інтерсептор та використовуємо його для взаємодії з сервером.

Нижче наведено код налаштування інтерсептора у файлі frontend/src/plugins/api.js.

```
import axios from 'axios';

const apiClient = axios.create({
baseURL: 'http://localhost:8000/'
});

// Add a request interceptor
apiClient.interceptors.request.use(
function (config) {
const token = localStorage.getItem('jwtToken');
if (token) {
config.headers.Authorization = `Bearer ${token}`;
}
return config;
},
function (error) {
return Promise.reject(error);
}
);

export default apiClient;
```

Цей код створює екземпляр Axios для взаємодії з API, встановлює базовий URL для запитів на локальний сервер за адресою http://localhost:8000/, додає інтерсептор запитів для додавання токену авторизації у заголовки, отриманий з localStorage, та експортує створений екземпляр axiosClient для використання в інших частинах застосунку.

Шаблонізатор у vue.js

Шаблонізатор у Vue.js відповідає за створення та відображення вмісту, який використовується для створення динамічних компонентів на сторінці. В основному файлі застосунку (App.vue або інший головний компонент) ми

підключаємо шаблонізатор та використовуємо його для відображення різноманітного контенту, такого як текст, зображення, кнопки та інше.

На рис. 1.17 відображенко створений шаблонізатор. Скриншот зроблений в безкоштовному редакторі коду Microsoft Visual Studio Code.

Рисунок 1.17. Шаблонізатор

Нижче наведено приклад коду для файлу шаблонізатора у файлі footer.vue.

```
<template>
    <footer class="text-light bg-light p-3">
        <div class="container">
            <div class="row row-cols-1 row-cols-sm-1 row-cols-md-3 row-cols-lg-4
row-cols-xl-4 row-cols-xxl-4 text-center justify-content-center align-items-center m-2">
                <div class="col m-1"><router-link class="link-dark text-
decoration-none" to="/">
                    <div data-bss-hover-animate="pulse">
                        <div class="row row-cols-1 text-center justify-
content-center align-items-center m-2">
                            <div class="col m-1"><i class="fas fa-book-open-
reader m-1 display-1"></i></div>
                            <div class="col m-1">
                                <h3 class="text-dark m-
1"><strong>Платформа</strong></h3>
                                <h6 class="text-black-50 m-1">з онлайн кусрів
для навчання та перевірки знань</h6>
                            </div>
                        </div>
                    </div>
                </router-link></div>
                <div class="col m-1">
                    <nav class="navbar navbar-light navbar-expand border-0">
                        <div class="container-fluid">
                            <div class="collapse navbar-collapse" id="navcol-2">
                                <ul class="navbar-nav d-flex flex-column align-
items-center align-content-center align-self-center m-auto justify-content-xxl-center">
                                    <li class="nav-item m-1" data-bss-hover-
animate="pulse"><router-link class="nav-link" to="/">Головна</router-link></li>
                                    <li class="nav-item m-1" data-bss-hover-
animate="pulse"><router-link class="nav-link" to="/signup">Рєстрація користувча</router-
link></li>
                                    <li class="nav-item m-1" data-bss-hover-
animate="pulse"><router-link class="nav-link" to="/signin">Авторизація
користувча</router-link></li>
                                    <li class="nav-item m-1" data-bss-hover-
animate="pulse"></li>
                                    <li class="nav-item m-1" data-bss-hover-
animate="pulse"><router-link class="nav-link" to="/courses">Навчальні курси та
тренажери</router-link></li>
```

```
                                        <li class="nav-item m-1" data-bss-hover-animate="pulse"><router-link class="nav-link" to="/cabinet">Щоденник користувача</router-link></li>
                                    </ul>
                                </div>
                            </div>
                        </nav>
                    </div>
                    <div class="col m-1"><router-link class="link-secondary text-decoration-none" to="/" target="_blank">
                                <p class="text-black-50 m-1" data-bss-hover-animate="pulse"><i class="fas fa-copyright me-1"></i>Платформа з онлайн кусрів для навчання та перевірки знань, 2024</p>
                            </router-link></div>
                </div>
            </div>
        </footer>
    </template>

    <script>
    export default
    {
      name: 'FooterComponent'
    }
    </script>

    <style scoped lang="scss">

    </style>
```

Цей код визначає шаблон для підвалу веб-сайту, який включає логотип та навігаційне меню. В шаблоні використовуються різні класи та стилі для оформлення та розташування елементів. Використовуються деякі вбудовані компоненти Vue.js, такі як <router-link>, для створення посилань та навігації. Створюється компонент Vue.js з ім'ям "FooterComponent". Шаблон має структуру з трьох основних розділів: логотип, навігаційне меню та інформація про авторські права. У шаблоні використовуються динамічні класи та анімації для покращення візуального вигляду. Використовується атрибут scoped для обмеження області дії стилів тільки для цього компонента.Цей компонент може бути використаний у головному файлі застосунку для відображення підвалу на кожній сторінці.

Однофайлові компоненти у Vue.js

Однофайлові компоненти у Vue.js забезпечують структуру, що дозволяє легко розробляти та підтримувати компоненти з розподілом логіки, розмітки та стилів в одному файлі. В основному файлі застосунку (App.vue або інший

головний компонент) ми підключаємо однофайлові компоненти та використовуємо їх для створення різноманітного контенту, такого як текст, зображення, кнопки та інше. Однофайлові компоненти Vue.js дозволяють зосередити все, що пов'язано з конкретним компонентом, в одному файлі, що полегшує розробку та підтримку коду. Це включає розмітку (HTML), логіку (JavaScript) та стилі (CSS).

На рис. 1.18 відображенко створені компоненти vue. Скриншот зроблений в безкоштовному редакторі коду Microsoft Visual Studio Code.

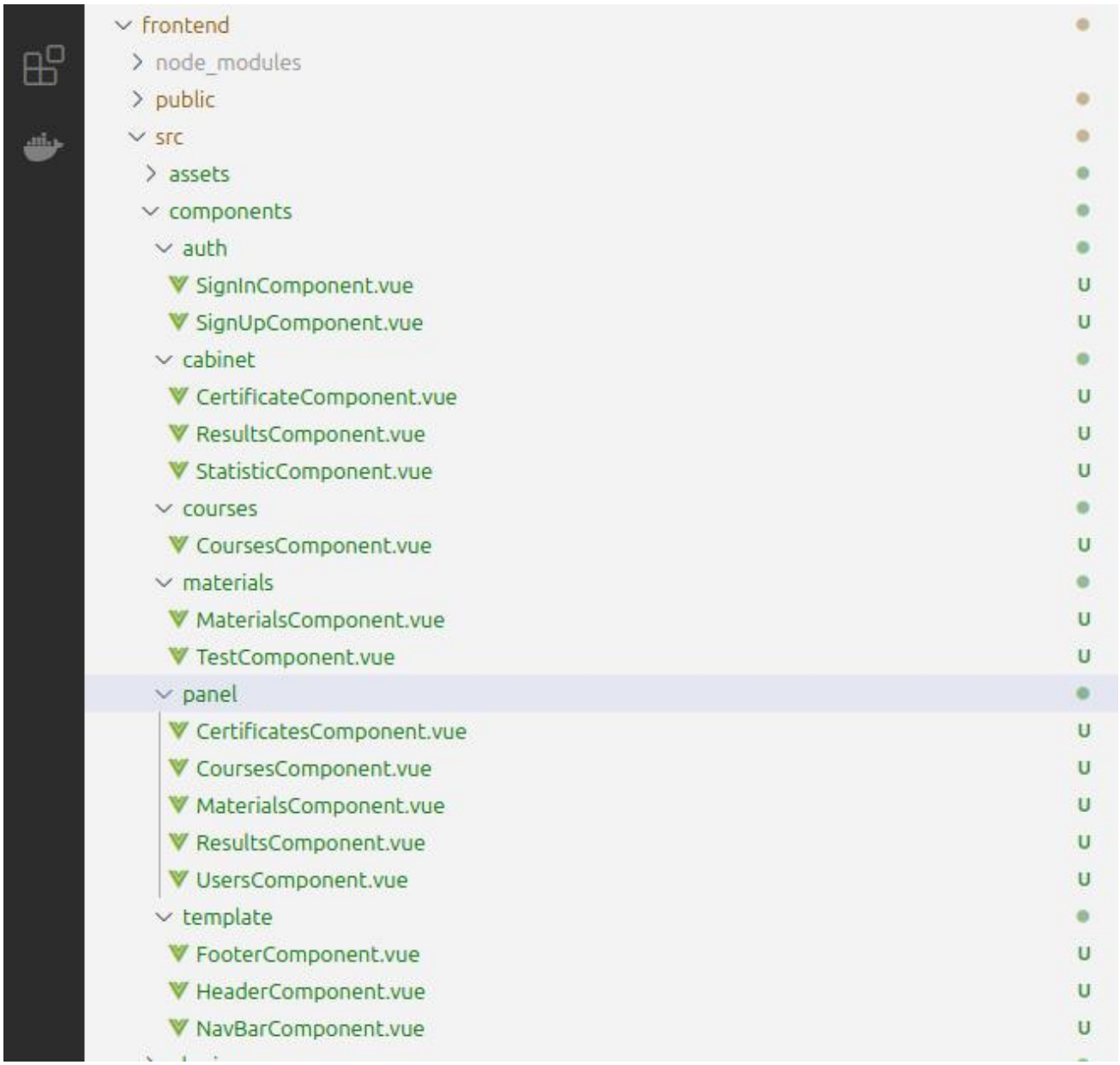


Рисунок 1.18. Компоненти vue

Нижче наведено код для однофайлового компонента виведення матеріалів курсу у файлі frontend/src/components/materials/MaterialsComponent.vue.

```
<template>
    <section class="bg-light">
        <div class="container p-3">
            <div class="row row-cols-1 justify-content-center align-items-center">
                <div class="col">
                    <div class="text-center m-3">
                        <h1 class="text-dark">Матеріали до теми</h1>
                        <p class="text-black-50">перелік матеріалів до теми</p>
                    </div>
                </div>
                <div class="col">
                    <div    class="card    m-3"    v-for="(item,    index)    in
data.data" :key="index">
                        <div class="card-body">
                            <h5                                          class="card-
title"><strong>{{ item.topic }}</strong></h5>
                            <p class="card-text">{{ item. comment }}</p>
                            <div  class="row  justify-content-center  align-items-
center">
                                <div class="col">
                                    <button    class="btn    btn-primary    w-100"
@click="pdfConverter(item.pdf)"                data-bss-hover-animate="pulse">Навчальний
матеріал</button>
                                </div>
                                <div class="col">
                                    <router-link    :to="'/test/'    +    item.id"
class="btn btn-primary w-100" data-bss-hover-animate="pulse">Тест</router-link>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>
</template>

<script>
export default
{
    name: 'MaterialsComponent',
    data: function()
    {
        return {
            url: 'api/materials/',
            data: []
        }
    },
    methods:
    {
        pdfConverter: function(pdfData)
        {
            const byteArray = new Uint8Array(pdfData.data);
            const blob = new Blob([byteArray], { type: 'application/pdf' });
            const pdfUrl = URL.createObjectURL(blob);
            window.open(pdfUrl);
```

```
            },
            getData: async function()
            {
                await this.$api.get(this.url + this.$route.params.id)
                .then((response) =>
                {
                    this.data = response.data;
                })
                .catch((error) =>
                {
                    this.status = error.response.data.message;
                });
            },
        },
        mounted()
        {
            this.getData();
        },
        unmounted()
        {

        }
    }
    </script>
```

Цей код у Vue.js створює компонент "MaterialsComponent.vue", який відображає перелік навчальних матеріалів. У шаблоні компонента визначено структуру сторінки з використанням класів Bootstrap для стилізації та створення адаптивного дизайну. Компонент завантажує дані про матеріали з API за допомогою методу getData, який викликається при монтуванні компонента, і зберігає ці дані у змінній data. Кожен матеріал відображається у вигляді картки з назвою, описом, кнопкою для завантаження PDF та посиланням на тест. Метод pdfConverter перетворює дані PDF у формат, який можна відкрити у новій вкладці браузера. Компонент також використовує Vue Router для створення посилань на тести, пов'язані з матеріалами. Таким чином, цей компонент забезпечує зручне відображення та взаємодію з навчальними матеріалами на веб-сторінці.

Представлення у Vue.js

Представлення (views) у Vue.js забезпечують структуру, що дозволяє легко розробляти та підтримувати різні частини інтерфейсу користувача, зосереджуючи логіку, розмітку та стилі в одному файлі. В основному файлі застосунку (App.vue або інший головний компонент) ми підключаємо представлення та використовуємо їх для створення різноманітного контенту,

такого як текст, зображення, кнопки та інше. Представлення Vue.js дозволяють зосередити все, що пов'язано з конкретним інтерфейсом, в одному файлі, що полегшує розробку та підтримку коду. Це включає розмітку (HTML), логіку (JavaScript) та стилі (CSS).

На рис. 1.19 відображенко створені представлення Vue. Скриншот зроблений в безкоштовному редакторі коду Microsoft Visual Studio Code.

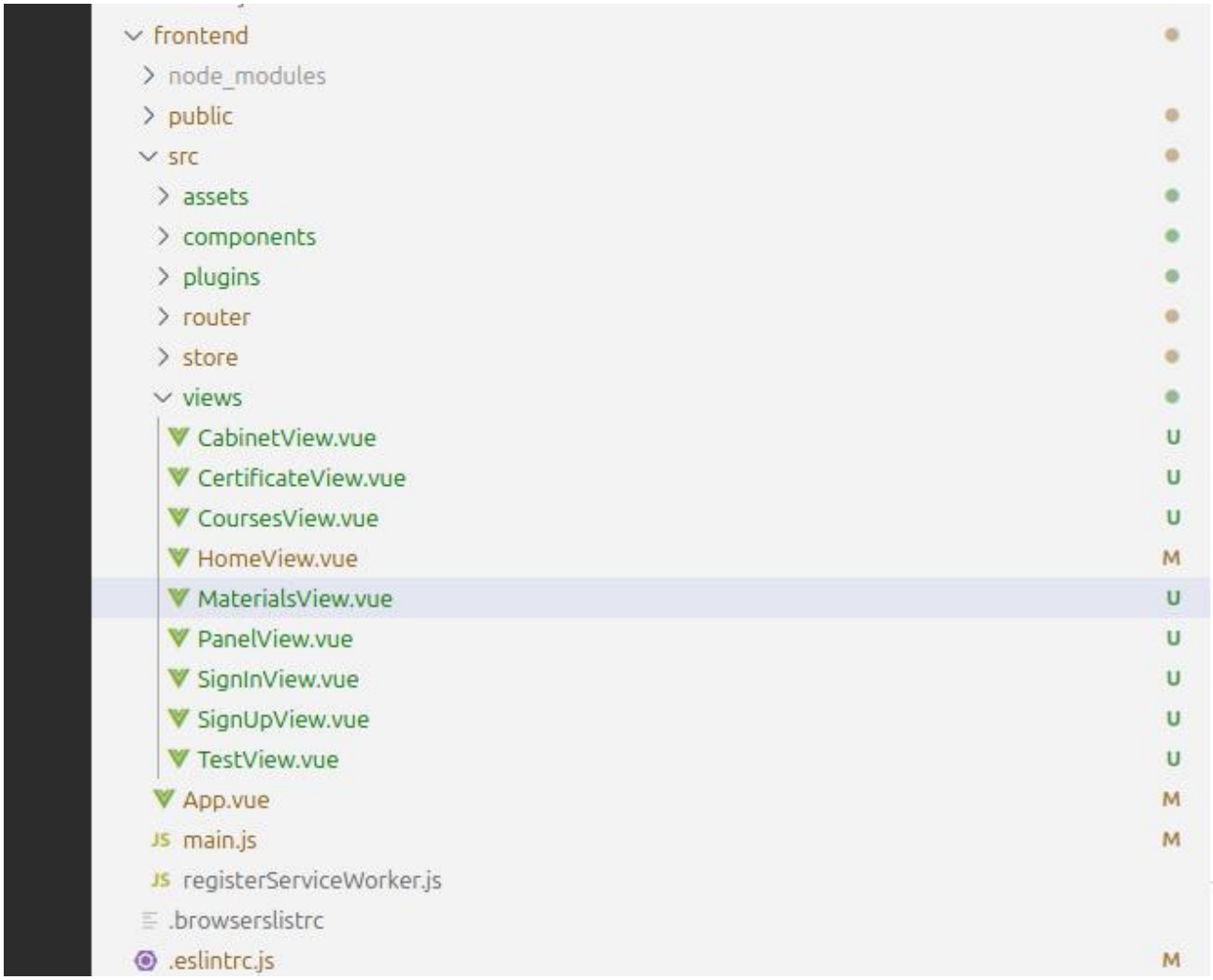


Рисунок 1.19. Представлення Vue

Нижче наведено код для представлення виведення матеріалів курсу у файлі frontend/src/components/materials/MaterialsView.vue.

```
<template>
<NavBarComponent/>

<MaterialsComponent/>

<FooterComponent/>
</template>
```

```
<script>
import NavBarComponent from '@/components/template/NavBarComponent.vue'
import MaterialsComponent from '@/components/materials/MaterialsComponent.vue'
import FooterComponent from '@/components/template/FooterComponent.vue'

export default
{
    name: 'MaterialsView',
    components:
    {
        NavBarComponent,
        MaterialsComponent,
        FooterComponent
    },
}
</script>
```

Цей код у Vue.js створює компонент "MaterialsView.vue", який використовується для відображення матеріалів курсу. У шаблоні компонента визначено три вкладені компоненти: NavBarComponent, MaterialsComponent та FooterComponent, які відповідають за навігаційну панель, контент з матеріалами та футер відповідно. Компонент імпортує ці вкладені компоненти з відповідних файлів. У секції script компонент зареєстрований під назвою MaterialsView та підключає вкладені компоненти до свого складу через об'єкт components. NavBarComponent відповідає за відображення навігаційної панелі зверху сторінки. MaterialsComponent відповідає за відображення основного контенту, тобто списку матеріалів. FooterComponent відповідає за відображення футера внизу сторінки, що забезпечує консистентний вигляд сторінки з матеріалами.

1.4 Тестування створеної платформи (QA)

Тестування API засобами Chrome DevTools дозволяє розробникам перевіряти запити та відповіді серверів безпосередньо в браузері, що забезпечує зручність і швидкість. Воно дає можливість бачити, які дані надсилаються та отримуються, включаючи заголовки, статуси відповідей і тіла запитів. Це дозволяє тестувати правильність роботи API, його швидкість і відповідність до документації. Крім того, розробники можуть імітувати різні сценарії, такі як відправка POST-запитів з різними параметрами або перевірка обробки помилок. Chrome DevTools також дозволяє відслідковувати мережеву активність в

реальному часі, що допомагає ідентифікувати та виправляти проблеми швидше. Інструмент Network у DevTools особливо корисний для виявлення проблем з CORS та іншими мережевавими помилками. Зрештою, використання Chrome DevTools для тестування API підвищує ефективність розробки та забезпечує надійність веб-застосунків.

На рис. 1.20 відображенко процес тестування API.

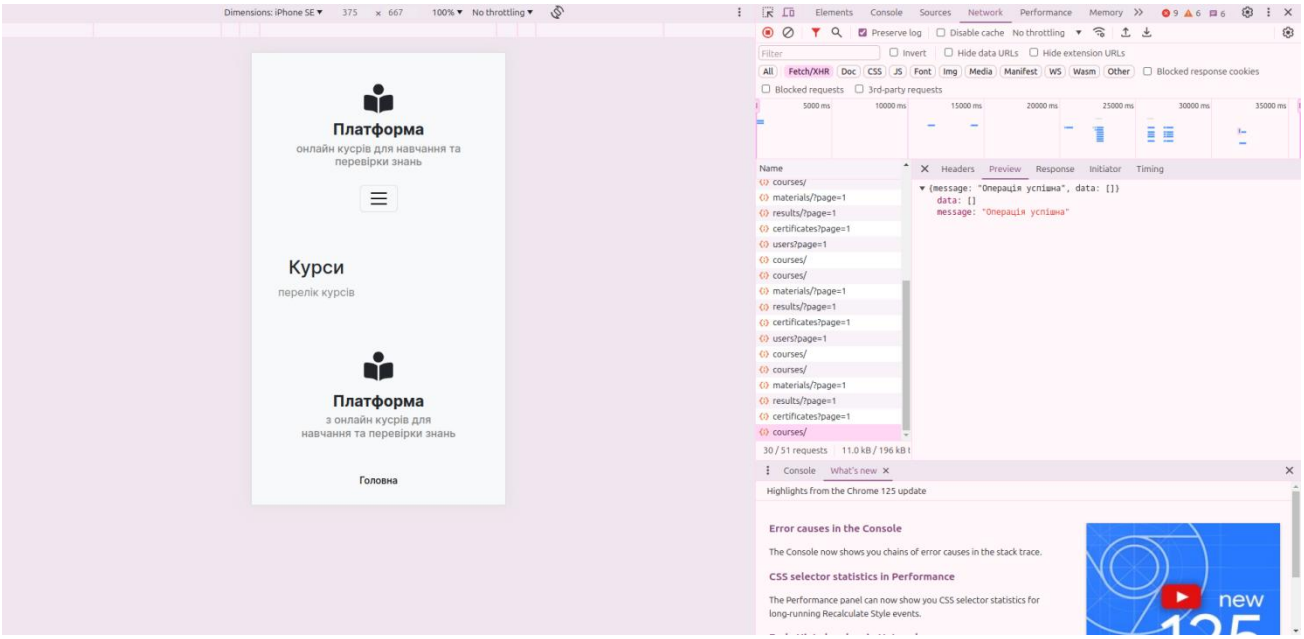


Рисунок 1.20. Тестування API

1.5 Огляд створеної платформи

1.5.1 Початкова сторінка

Початкова сторінка представляє собою веб-сторінку, котру бачить користувач під час завантаження веб-застосунку. Ця веб-сторінка містить промо-матеріал навчальної платформи.

На рис. 1.21 відображено початкову сторінку навчальної платформи з промо-матеріалом.

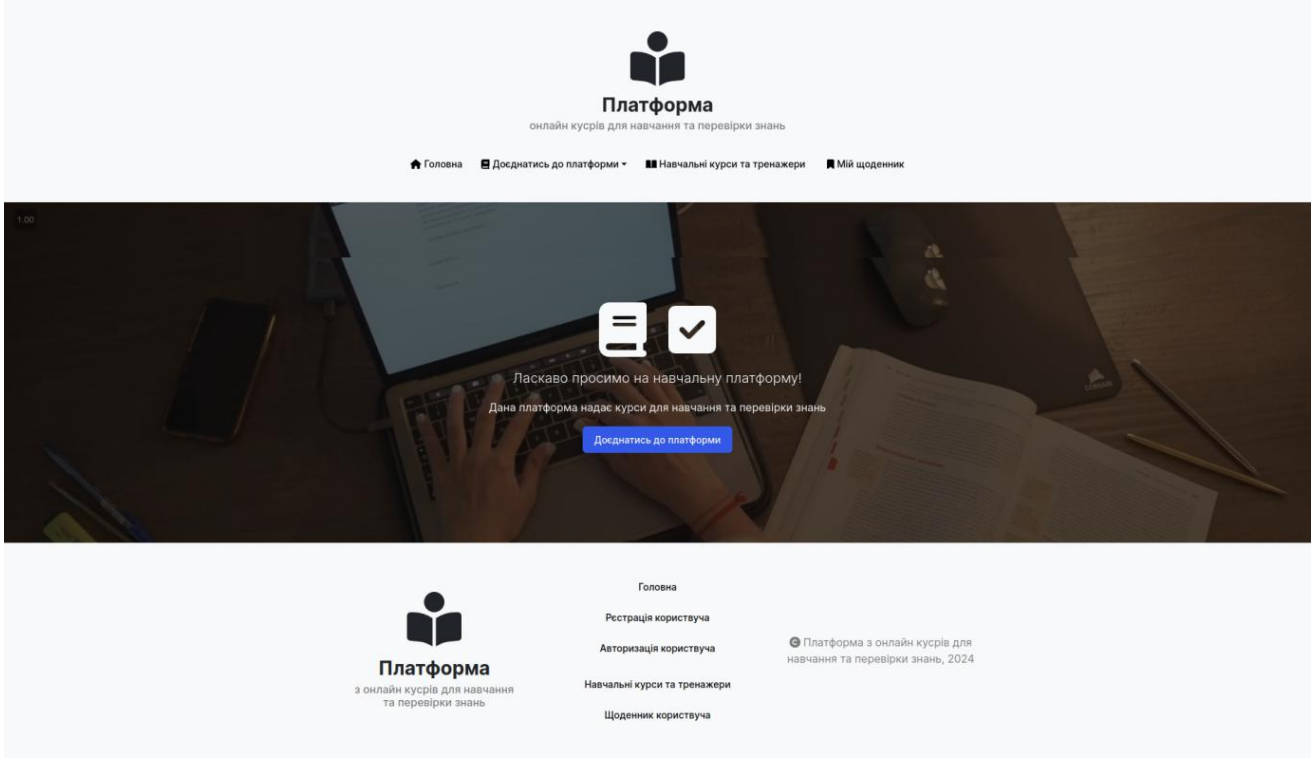


Рисунок 1.21. Початкова

1.5.2 Авторизація та реєстрація

Веб-сторінка реєстрації.

Реєстрація представляє собою процес створення нового облікового запису користувача в системі. На веб-сторінці реєстрації користувачі зазвичай повинні ввести свої особисті дані, такі як ім'я, прізвище, адреса електронної пошти тощо, а також обрати унікальний логін та пароль для майбутнього входу в систему. Після успішної реєстрації користувач отримує можливість авторизуватися та використовувати ресурси платформи у майбутньому. Обидва ці процеси, авторизація та реєстрація, є ключовими для забезпечення безпеки та зручності використання навчальної платформи користувачами.

На рис. 1.22 відображено сторінку реєстрації на навчальній платформі.

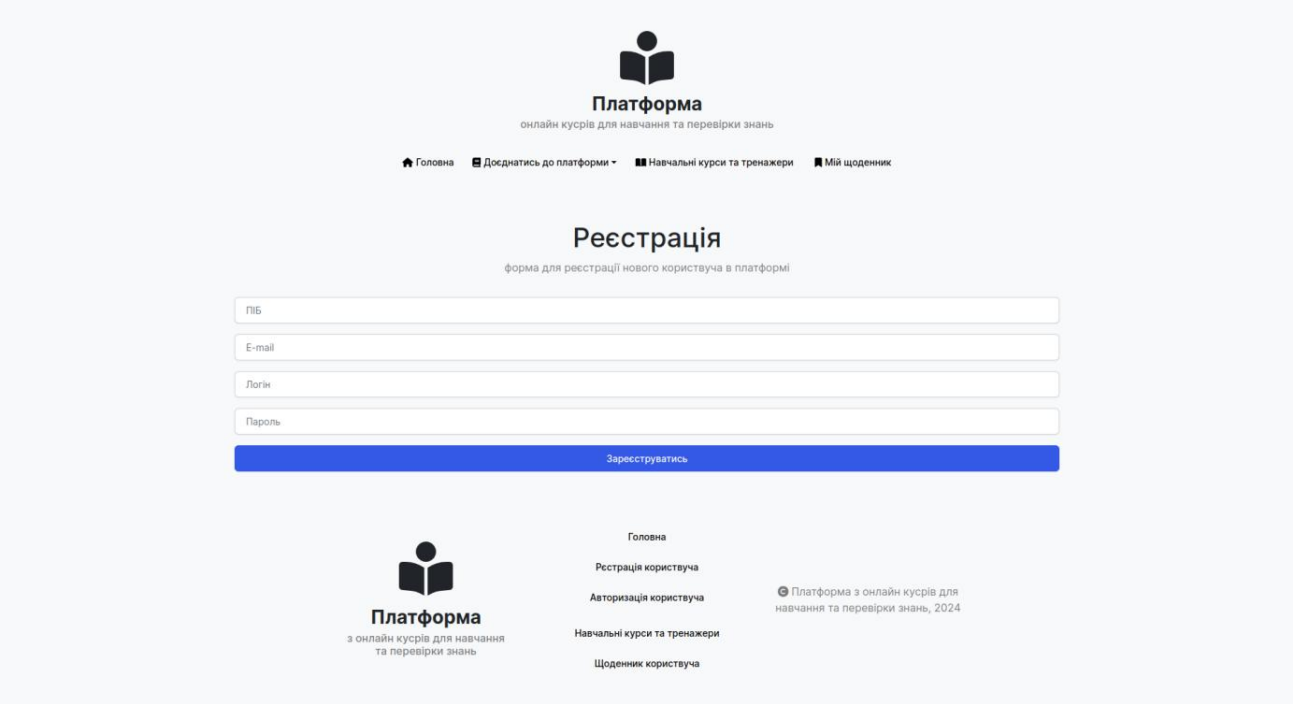


Рисунок 1.22. Реєстрація

Веб-сторінка авторизації.

Авторизація представляє собою процес перевірки ідентифікації користувача перед наданням доступу до до ресурсу та функцій платформи. На навчальній платформі авторизація виконується шляхом введення користувачем свого логіну (ідентифікатора) та пароля.

На рис. 1.23 відображено сторінку авторизації на навчальній платформі.

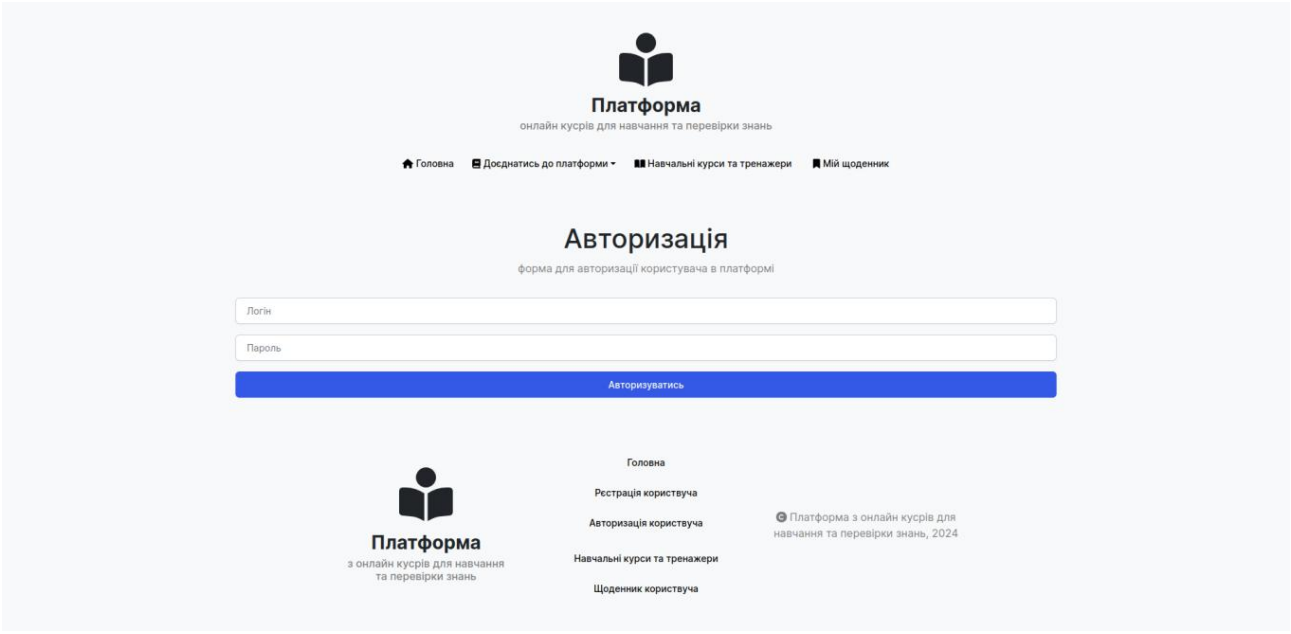


Рисунок 1.23. Авторизація

1.5.3 Кабінет користувача та адміністратора

Веб-сторінка з кабінетом користувача.

У кабінеті користувача на навчальній платформі доступні інструменти для відстеження успішності та оцінок у навчальних курсах. Користувач може переглядати свій прогрес у кожному курсі, перевіряти власні оцінки за завдання та тести, а також отримувати звіти про виконані дії. Такий кабінет надає користувачеві повний контроль над його академічною діяльністю та допомагає зосередитися на досягненні навчальних цілей.

На рис. 1.24 відображено веб-сторінку з кабінетом користувача на навчальній платформі.

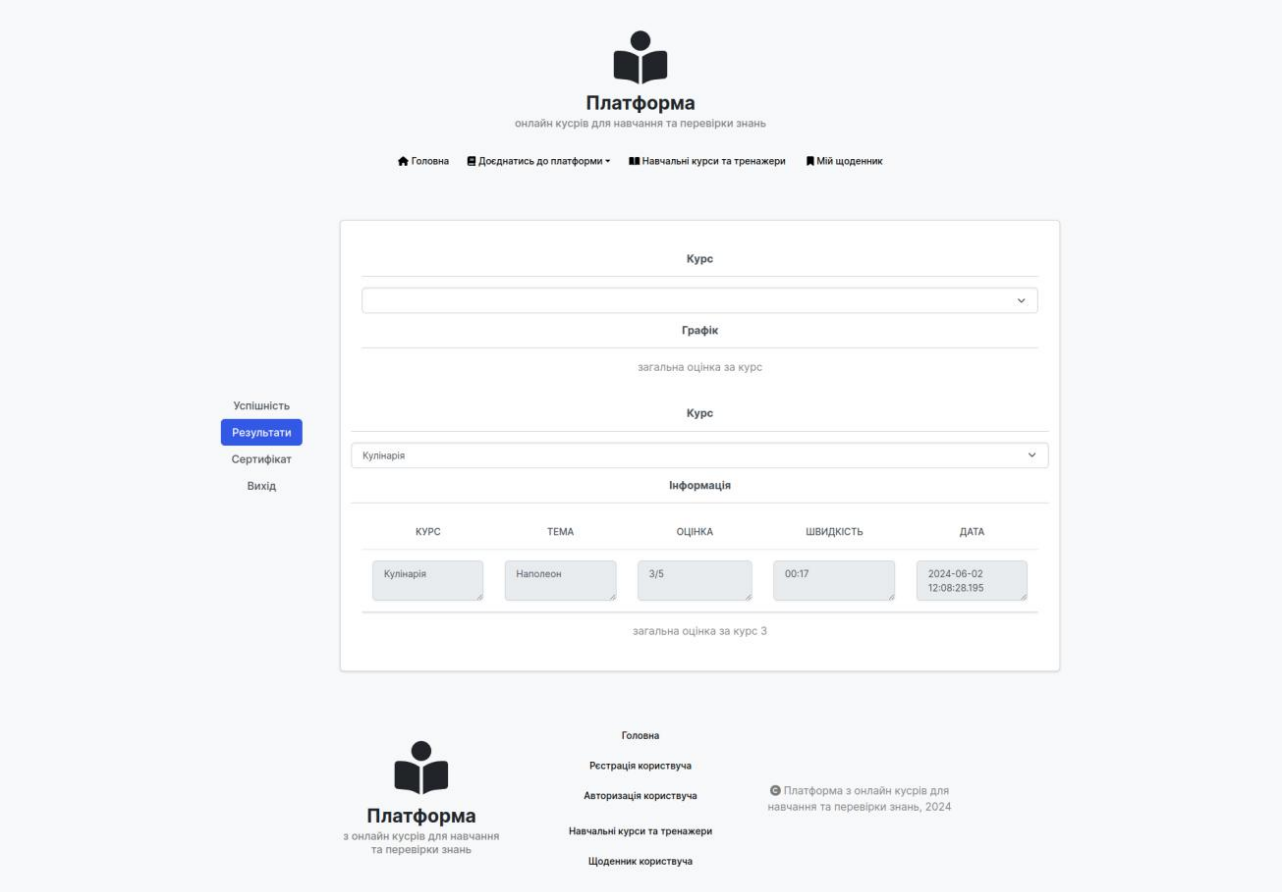


Рисунок 1.24. Кабінет користувача

Веб-сторінка з кабінетом адміністратора.

Кабінет адміністратора на навчальній платформі є центром управління контентом та курсами. У цьому кабінеті адміністратор може створювати нові

курси, редагувати наявні, а також видаляти їх за необхідності. Крім того, адміністратор має можливість додавати різноманітний навчальний матеріал до кожного курсу, такий як лекції, тестові завдання тощо. Кабінет адміністратора забезпечує зручний і ефективний спосіб організації навчального процесу на платформі.

На рис. 1.25 відображено веб-сторінку з кабінетом адміністратора на навчальній платформі.

Рисунок 1.25. Кабінет адміністратора

1.5.4 Курси та матеріали

Веб-сторінка з курсами.

Сторінка з переліком курсів на навчальній платформі відображає доступні для вивчення навчальні програми та матеріали. Користувачі можуть переглядати список курсів, вибирати ті, які їх цікавлять, та отримувати докладну інформацію

про кожен курс. Ця сторінка дозволяє користувачам швидко знайти і обрати навчальні програми, які найкраще відповідають їхнім потребам та інтересам.

На рис. 1.26 відображено веб-сторінку зі створеними курсами на навчальній платформі.

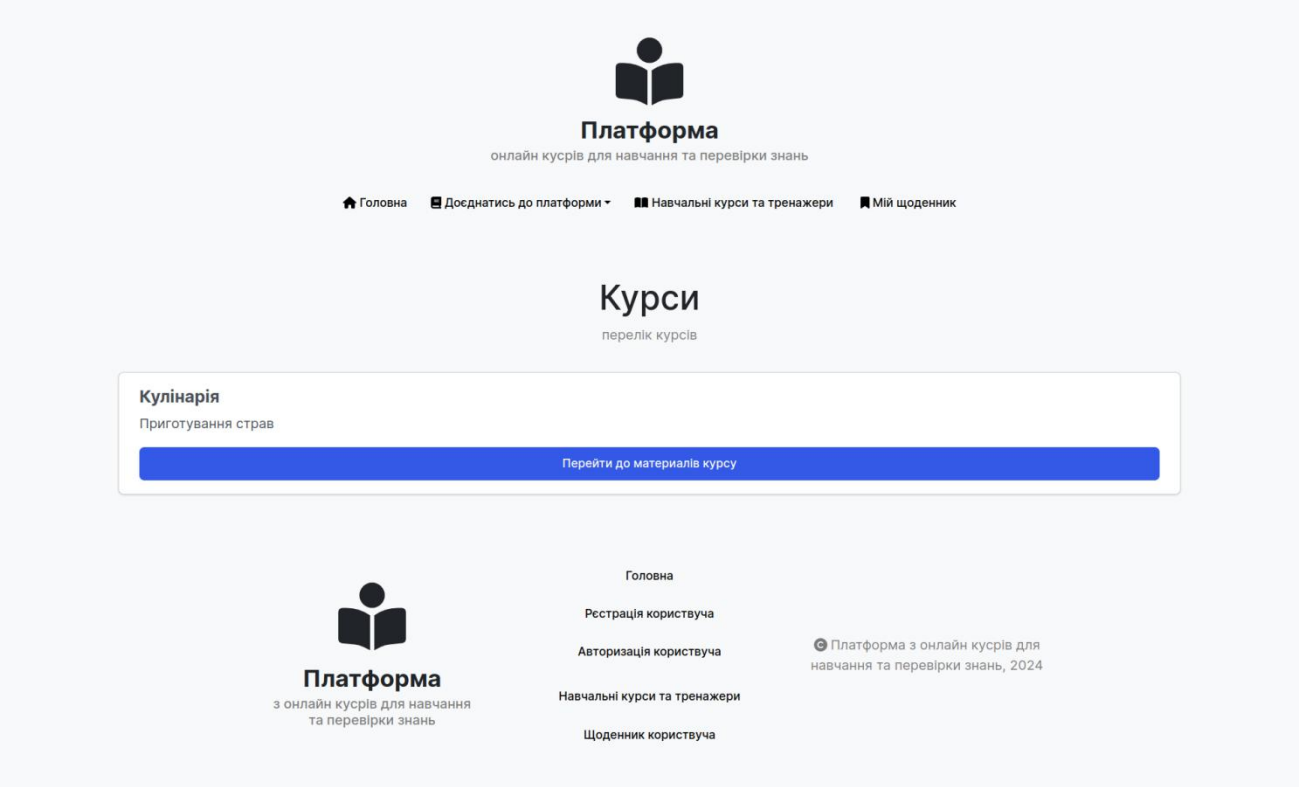


Рисунок 1.26. Курси

Веб-сторінка з матеріалами.

На сторінці з матеріалами обраного курсу користувачі знаходять всі необхідні ресурси для вивчення конкретної теми. Тут доступні лекції, завдання для практики. Ця сторінка створена для зручного та систематичного ознайомлення з курсовим матеріалом та підвищення ефективності навчання.

На рис. 1.27 відображено веб-сторінку з матеріалами курсу на навчальній платформі.

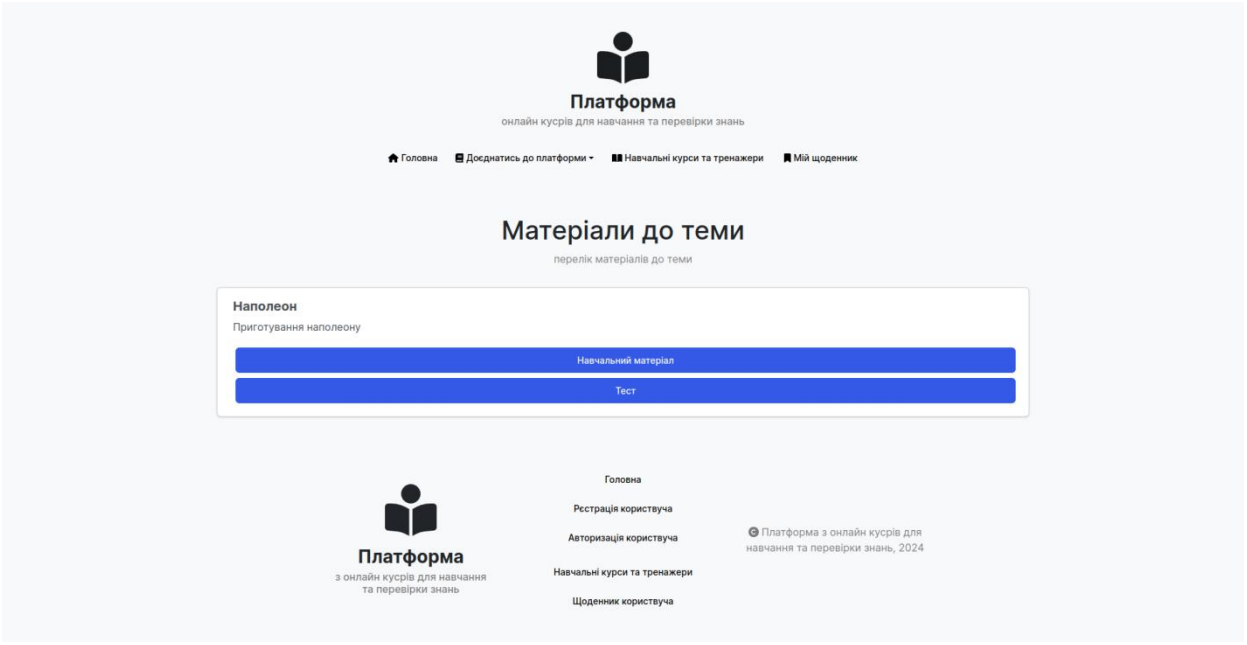


Рисунок 1.27. Матеріали курсу

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проєкті розроблено Веб-орієнтована платформа для навчання та перевірки знань. Її використання є ключовим для оптимізації навчальних процесів, надаючи можливість швидкого реагування на зміни в освітньому середовищі та підвищення рівня якості навчання.

Для розробки програмного продукту використовувалися передові технології, зокрема база даних SQLite та мова програмування JavaScript з використанням платформи Node.js. Цей інструментарій забезпечив високий рівень ефективності та продуктивності, відповідаючи вимогам сучасного ринку навчальних технологій.

Платформа має розширений функціонал, включаючи систему управління курсами, матеріалами та інші важливі компоненти для організації навчального процесу.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення. Оцінка якості програмного продукту включає визначення трудомісткості та вартості його створення.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, усл. машинних командах**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП введення інформації	1060 – 5750

У табл. 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл. 2.2.

Таблиця 2.2. Обсяг ПП

Обсяг ПП, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк = 0,7 ÷ 0,8): Тар = 229 х 0,7 = 160,3 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \leftrightarrow L_1 \leftrightarrow K_H \quad (2.1)$$

$$T_{ТП} = T^a p \leftrightarrow L_2 \leftrightarrow K_H \quad (2.2)$$

$$T_{РП} = T^a p \leftrightarrow L_3 \leftrightarrow K_H \leftrightarrow K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5).

Таблиця 2.3.Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	**Ступінь новизни**		
	А	Б	В
ТЗ (L1)	0,15	0,12	0,12
ТП (L2)	0,16	0,15	0,11
РП (L3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
А	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПО маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	**Значення Кт**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання:

$$T_{тз}=Ta*L_1*K_н=160,3 \ *0,12*0,7= 13,46 \text{ (люд/годин)} \qquad (2.1)$$

Трудомісткість розробки технічного проекту:

$$T_{тп}=Ta*L_2*K_н=160,3 \ *0,11*0,7 = 12,34 \text{ (люд/годин)} \qquad (2.2)$$

Трудомісткість розробки робочого проекту:

$$T_{рп}=Ta*L_3*K_н*K_т=160,3 \ *0,61*0,7*0,8= 54,75 \text{ (люд/годин)} \qquad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$=2 (стор), розробка ТП $N_{тп}$=43 (стор), розробка робочого проєкту $N_{рп}$=10 (стор), пояснювальна записка відповідно $N_{пз}$=25 (стор). Розрахунок зведений у табл. 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин**		
1.ТЗ	ТРтз=13,46	$T_{кк}=0,7*N_{тз}=$ 0,7*2=1,4	$T_{нк}=0,15*N_{тз}=$ 0,15*2=0,30
2.Розробка ТП	ТРтп=12,34	$T_{кк}=0,7*N_{тп}=$ 0,7*43=30,1	$T_{нк}=0,15*N_{тп}=$ 0,15*43=6,45
3.Розробка РП	ТРрп= 54,75	$T_{кк}=0,7*N_{рп}=$ 0,7*10=7,0	$T_{нк}=0,15*N_{рп}=$ 0,15*10=1,5
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}=$ 1,5*25 =37,5	$T_{кк}=0,7*N_{тз}=$ 0,7*25=17,5	$T_{нк}=0,15*N_{пз}=$ 0,15*25 =3,75
Усього, в т.ч.:	213,53		
- на розробку	$T_р$=118,05		
- контроль керівника		$T_{кк}$=56	
- нормоконтроль			$T_{нк}$=12

2.3 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, загальні витрати на розробку ПП. Розрахунок основної заробітної плати виконавців приведений у табл. 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2024» встановлено мінімальну заробітну плату у місячному розмірі з 1 квітня 2024 року - 8000 гривень; мінімальну погодинну тарифну ставку – 46,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	118,05	55,00	6492,75
2.Контроль керівника	56	120,00	6720,00
3.Нормоконтроль	12	100,00	1200,00
Усього	-	-	Зо= 14412,75

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в табл. 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	80	4.0	320,0
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0,1 \leftrightarrow Вм1 =$ 0,1*320 =32,00
Усього				$В_м = В_{мі} + В_{тр\_з} =$ 352,00

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в табл. 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	352,00	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	14412,75	$З_о$ (див. табл. 2.7)
3.Додаткова заробітна плата	1441,28	$Зд = 0,1 \leftrightarrow Зо$ = 14412,75*0,1
4.Відрахування до єдиного фонду соціального внеску	3487,88	$Вє.с.в. = 0,22 \leftrightarrow (Зо + Зд)$ =0,22* (14412,75+1441,28)
5. Накладні витрати	5765,10	$Внак. = 0,4 \leftrightarrow Зо$ =0,4 * 14412,75
6. Повна собівартість	25459,08	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 352,00+14412,75+1441,28+3487,88+5765,10

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_п*Р)/100=(25459,08*10)/100=2545,91 грн. \quad (2.4)$$

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о = С_п+П=25459,08+2545,91=28004,99 грн. \quad (2.5)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$$Ц_р=Ц_о+ПДВ=28004,99 +28004,99 *0.2=33605,98 грн. \quad (2.6)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

3.1 Вступ

Основному закону держави – Конституції України повинні базуватися і відповідати всі закони і підзаконні акти держави. Конституція України прийнята Верховною Радою 26 червня 1996 року. В ній декларуються права і свобода всіх громадян України. Ці права і свобода для сфери трудової діяльності конкретизовані в законах України і нормативно-правових актах про охорону праці (НПАОП), Державних стандартах та постановах Кабінету Міністрів України, що стосуються охорони праці.

Забезпечення здорових і безпечних умов праці покладається на адміністрацію підприємств, установ, організацій. Вона зобов'язана впроваджувати сучасні засоби техніки безпеки, які попереджують виробничий травматизм і забезпечують санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань.

В даному розділі дипломного проєкту розглядається питання охорони праці на робочому місці програміста.

3.2 Аналіз умов праці

Під час виконання роботи програмісти зіштовхуються із впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика тощо.

3.3 Гігієнічні вимоги до виробничого середовища

На робочому місці програміста повинні бути створені умови для безпечної та високопродуктивної праці.

3.3.1 Вимоги до приміщення

Розміщення робочих місць з персональним комп'ютером у підвальних приміщеннях, на цокольних поверхах заборонено. Об'ємно-планувальні рішення будівель та приміщень для роботи з ВДТ мають відповідати вимогам ДСанПіН 3.3.2.007-98. Площа на одне робоче місце становить не менше 6,0 м2, а об'єм – не менше ніж 20,0 м3. У приміщеннях слід щоденно робити вологе прибирання, повинні бути оснащені аптечками першої медичної допомоги. При приміщеннях мають бути обладнанні побутові приміщення для відпочинку.

3.3.2 Виробниче освітлення

У приміщеннях, призначених для роботи з відео терміналами повинні бути природне та штучне освітлення, доцільно, щоб вікна були орієнтовані на північ або північний захід. Що регулювати рівень освітленості і захищати від прямого влучення сонячних променів на робоче місце, на вікнах повинні бути штора або жалюзі. При кольоровому оформленні виробничих і допоміжних приміщень необхідно враховувати орієнтацію їхніх вікон стосовно частин світу і використовувати гармонійне сполучення кольорів. Стіни і робочі поверхні кольори мають мало насичені (основні) , для невеликих помешкань або ділянок, що рідко потрапляють у поле зору працюючих, а також для створення контрастності – кольори середньої насиченості (допоміжні), для маленьких по площі поверхонь – насичені (акценти) – як функціональне фарбування. Стелі у всіх приміщеннях повинні бути білими. Для виключення випадку відблисків світла в очі працюючого поверхні устаткування в приміщеннях повинні бути матовими , а стіни бути пофарбованими фарбами пастельних тонів.

У приміщенні для штучного освітлення використовуються люмінесцентні лампи типу ЛБ, які в порівнянні з лампами розжарювання мають ряд істотних переваг: за спектральним складом світла вони близькі до природного світла, мають підвищену світлову віддачу (у 2-5 разів вищу, ніж у ламп розжарювання); мають триваліший термін служби – до 10 тис годин. У світильниках місцевого освітлення допускається застосування ламп розжарювання.

3.3.3 Гігієнічні нормування параметрів повітря робочої зони

Відповідно до ГОСТ 12.1.005-88, СН 4088-86 у виробничих приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря.

У таблиці 3.1 надано параметри мікроклімату.

Таблиця 3.1. Параметри мікроклімату

Параметри мікроклімату	**Значення параметрі**	**значення параметрі**
	Взимку	влітку
Температура, C^0	22-24	23-25
Відносна вологість, %	40-60	40-60
Швидкість руху повітря, м/с	0,1	0,1-0,2

Для підтримки в приміщеннях нормального, що відповідає гігієнічним вимогам складу повітря, видалення з нього шкідливих газів, пилу використовують вентиляцію. При природній вентиляції (за допомогою вікон) повітря надходить у приміщення і видаляється з нього внаслідок різниці температур і тиску. Механічна вентиляція забезпечується вентиляторами, кондиціонерами.

3.3.4 Організація робочих місць

Робочі місця з ПК доцільно розміщати в глибині приміщення. Розташування відео термінала, при якому працюючий звернений обличчям або спиною до вікон, неприпустимо при будь-якому способі реалізації загального висвітлення, як прямим, так і відбитим світлом. Робочі місця повинні бути розташовані так, щоб у поле зору працюючого не попадали поверхні, що мають властивість віддзеркалювання, вікна освітлювальні прилади. встановлюватися під кутом 90-100 градусів від вікон, так, щоб світло падало з боку.

Рекомендовані розміри столу: висота 725 мм, ширина 600-1400 мм, глибина 800-1000 мм. Робочий стілець повинен бути оснащений підйомно-

поворотним пристроєм для регулювання висоти сидіння і спинки, а також кута її нахилу. Регулювання кожного параметра повинно проводитися легко, бути незалежним і надійно фіксуватися. Робочий стіл повинен регулюватися по висоті в границях 680-800 мм, а ширина – забезпечувати можливість виконання операцій в зоні досяжності моторного поля

Клавіатуру слід розташовувати на поверхні столу на відстані 100…300 мм від краю, звернутого до працюючого.

3.3.5 Електробезпека

Використання електричної енергії на виробництві повезене з небезпекою дії електроструму на організм людини. Дотик до оголених проводів,незаземлених металевих корпусах електричного обладнання при відкритих рубильниках може призвести до ураження електрострумом.

Для попередження поразок електричним струмом необхідно:

- У повному обсязі виконувати правила провадження робіт і правил технічної експлуатації;
- Виключати можливість доступу працівника до частин устаткування, що працює під небезпечною напругою, неізольованим частинам, призначеним для роботи при малій напрузі й не підключеним до захисного заземлення;
- Застосовувати ізоляцію, що служить для захисту від поразки електричним струмом.

У приміщенні, де одночасно експлуатуються понад п’ять ЕОМ, встановлюється на помітному та доступному місці аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ЕОМ з ВДТ і ПК повинні підключатися до електромережі тільки за допомогою справних штепсельних з’єднань і електророзеток заводського виготовлення.

Штепсельні з’єднання та електророзетки для напруги 12В та 42В за своєю конструкцією та візуально (за кольором) мають відрізнятися від штепсельних з’єднань для напруги 127В та 220В.

3.4 Пожежна безпека

Пожежі руйнують виробничі будівлі, знищують матеріали і готову продукцію, приводять в негідність обладнання, на тривалий час припиняють роботи в цехах.

Пожежна профілактика – це комплекс заходів, спрямованих на попередження пожежі, створення умов, сприяючих швидкій ліквідації пожежі. Заходи щодо пожежної безпеки підрозділяють на дві основні групи: попередження пожеж і ліквідація вже виниклих пожеж

Протипожежний захист приміщення забезпечується застосуванням автоматичної установки пожежної сигналізації, наявністю засобів пожежогасіння, застосуванням основних будівельних конструкцій будинку з регламентованими межами вогнестійкості, організацією своєчасної евакуації людей.

Для ліквідації пожеж використовують первинні засоби пожежогасіння, які призначенні для гасіння пожеж у початковій стадії їх розвитку. Вони є у всіх виробничих приміщеннях, цехах.

Оснащення об'єктів первинними засобами пожежогасіння проводиться відповідно до Правил пожежної безпеки в Україні,введених в дію наказом внутрішні справ України від 22.06.95 №400.

До первинних засобів пожежогасіння відносяться : вогнегасники, пожежний інвентар (покривала з негорючого теплоізоляційного полотна, грубововняної тканини або повсті, ящики з піском, бочки з водою, пожежні відра, совкові лопати) та пожежний інструмент (гаки, ломи, сокири тощо).

Бочки для зберігання води з метою пожежогасіння відповідно до ГОСТ 12.4.009-83 повинні мати місткість не менше 0.2м3 і бути укомплектованими пожежним відром місткістю не менше 0.008м3.

Пожежні щити (стенди) встановлюють на території об’єкта з розрахунку один щит (стенд) на площу 5000м2. Ящики для піску повинні мати місткість 0.5, 1.0 або 3.0м2 та бути укомплектованими совковою лопатою. Вмістилище для піску, що є елементом конструкції пожежного стенду, повинні бути місткістю не менше 0.1м3. Конструкція ящика (вмістилище) повинна забезпечувати зручність діставання піску та усування попадання опадів.

.

ВИСНОВКИ

Веб-орієнтована платформа для навчання та перевірки знань, розроблена у рамках дипломного проєкту, відповідає сучасним стандартам у галузі програмування та баз даних. Її використання є ключовим для оптимізації навчальних процесів, надаючи можливість швидкого реагування на зміни в освітньому середовищі та підвищення рівня якості навчання.

Для розробки програмного продукту використовувалися передові технології, зокрема база даних SQLite та мова програмування JavaScript з використанням платформи Node.js. Цей інструментарій забезпечив високий рівень ефективності та продуктивності, відповідаючи вимогам сучасного ринку навчальних технологій.

Платформа має розширений функціонал, включаючи систему управління курсами, матеріалами та інші важливі компоненти для організації навчального процесу.

У пояснювальній записці детально розглянуті всі аспекти, визначені у технічному завданні на дипломне проєктування. Проведено аналіз предметної області, описано використані технології та інструменти, розраховано економічну ефективність проєкту та розглянуто питання безпеки праці, наведений список використаних джерел.

Завершено, розроблений програмний продукт представляє собою ефективну платформу для навчання та перевірки знань, що відповідає сучасним вимогам освітньої галузі.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Udemy. [Веб-сайт]. URL: https://www.udemy.com/.
2. Coursera. [Веб-сайт]. URL: https://www.coursera.org/.
3. Prometheus. [Веб-сайт]. URL: https://prometheus.org.ua/.
4. StackOverflow. [Веб-сайт]. URL: https://stackoverflow.com/.
5. SQLite Docs. [Веб-сайт]. URL: https://www.sqlite.org/docs.html.
6. Express.js Docs. [Веб-сайт]. URL: https://expressjs.com/en/5x/api.html.
7. Vue.js Docs. [Веб-сайт]. URL: https://vuejs.org/guide/introduction.html.
8. Bootstrap Docs. [Веб-сайт]. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/.
9. В. В. Босько, Л. В. Константинова, К. М. Марченко, О. С. Улічев. Web-програмування. Частина 1 (Frontend): Навчальний посібник. – «Кропивницький центральноукраїнський національний технічний університет», 2022.
10. О. С. Бунке. Серверні WEB-технології: Навчальний посібник. – «КПІ ім. Ігоря Сікорського», 2023. 5. 5.С. І. Доценко. Організація та системи керування базами даних: Навчальний посібник. – «Український державний університетзалізничного транспорту», 2023.

ДОДАТОК А. Програмний код основної логіки веб-застосунку

```
// courses.js

// Create router
var express = require('express')
var router = express.Router()

// Include DB
var db = require("../utils/db")

// Include Auth
var auth = require("../utils/auth")

// Middleware for parsing data
var bodyParser = require("body-parser")
router.use(bodyParser.urlencoded({ extended: false }))
router.use(bodyParser.json())

// Get a List of courses
router.get("/api/courses", (req, res, next) => {
    if (auth.isUser(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var sql = "select * from courses"
    var params = []

    db.all(sql, params, (err, rows) => {
        if (err) {
          res.status(400).json({"error":err.message})
          return
        }
        res.json({
            "message":"Операція успішна",
            "data":rows
        })
      })
})

// Get a single Course by id
router.get("/api/courses/:id", (req, res, next) => {
    if (auth.isUser(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var sql = "select * from courses where id = ?"
    var params = [req.params.id]

    db.get(sql, params, (err, row) => {
        if (err) {
          res.status(400).json({"error":err.message})
          return
        }
        res.json({
            "message":"Операція успішна",
            "data":row
```

```
        })
    })
})

// Add a Course
router.post("/api/courses", (req, res, next) => {
    if (auth.isAdmin(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var data = {
        name: req.body.name,
        comment: req.body.comment
    }
    var sql = "INSERT INTO courses (name, comment) VALUES (?, ?)"
    var params = [data.name, data.comment]

    db.run(sql, params, function (err, result) {
        if (err) {
            res.status(400).json({"error": err.message})
            return
        }
        res.json({
            "message": "Операція успішна",
            "data": data,
            "id" : this.lastID
        })
    })
})

// Edit a Course by id
router.patch("/api/courses/:id", (req, res, next) => {
    if (auth.isAdmin(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var data = {
        name: req.body.name,
        comment: req.body.comment
    }
    var sql = `UPDATE courses set
    name = COALESCE(?,name),
    comment = COALESCE(?,comment)
    WHERE id = ?`
    var params = [data.name, data.comment, req.params.id]

    db.run(sql, params, function (err, result) {
        if (err) {
            res.status(400).json({"error": res.message})
            return
        }
        res.json({
            message: "Операція успішна",
            data: data,
            changes: this.changes
        })
    })
})

// Delete a Course by id
router.delete("/api/courses/:id", (req, res, next) => {
    if (auth.isAdmin(req.headers.authorization) == false) {
```

```
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var sql = "DELETE FROM courses WHERE id = ?"
    var params = [req.params.id]

    db.run(sql, params, function (err, Certificate) {
        if (err) {
            res.status(400).json({"error": res.message})
            return
        }
        res.json({"message":"Операція успішна", changes: this.changes})
    })
})

module.exports = router

// Materials.js

// Create router
var express = require('express')
var router = express.Router()

// Include DB
var db = require("../utils/db")

// Include Auth
var auth = require("../utils/auth")

// Include Test logic
var test = require("../utils/test")

// Middleware for parsing data
var bodyParser = require("body-parser")
router.use(bodyParser.urlencoded({ extended: false }))
router.use(bodyParser.json())

var fileUpload = require('express-fileupload');
router.use(fileUpload());

// Get a List of All Materials by Admin
router.get("/api/materials", (req, res, next) => {
    var perPage = 2 // number of items per page
    var page = req.query.page || 1 // current page (default to 1 if not provided)
    var offset = (page - 1) * perPage // calculate the offset for the SQL query

    var query = req.query.query || 'all'

    var sql = `SELECT *
    FROM materials`
    var params = []
    var countSql = `SELECT COUNT(*) as count FROM materials`
    var countParams = []

    if (query !== 'all') {
        sql += ` WHERE topic LIKE ?`
        countSql += ` WHERE topic LIKE ?`
        params.push(`%${query}%`)
        countParams.push(`%${query}%`)
    }

    sql += ` ORDER BY materials.id DESC`
```

```
    sql += ` LIMIT ? OFFSET ?`

    params.push(perPage, offset)

    db.get(countSql, countParams, (err, result) => {
        if (err) {
          res.status(400).json({ "error": err.message })
          return
        }

        var totalCount = result.count
        var totalPages = Math.ceil(totalCount / perPage)

            db.all(sql, params, (err, rows) => {
            if (err) {
                res.status(400).json({ "error": err.message })
                return
            }

            res.json({
                "message": "Операція успішна",
                "data": rows,
                "pagination": {
                    "currentPage": page,
                    "perPage": perPage,
                    "totalItems": totalCount,
                    "totalPages": totalPages
                }
            })
        })
    })
})

// Get a List of Materials by Course
router.get("/api/materials/:id", (req, res, next) => {
    if (auth.isUser(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var sql = `SELECT * FROM materials WHERE course = ?`
    var params = [req.params.id]

    db.all(sql, params, (err, rows) => {
        if (err) {
            res.status(400).json({ "error": err.message })
            return
        }

        res.json({
            "message": "Операція успішна",
            "data": rows
        })
    })
})

// Get a single Material by id
router.get("/api/test/:id", (req, res, next) => {
    if (auth.isUser(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var sql = "select * from materials where id = ?"
```

```
    var params = [req.params.id]

    db.get(sql, params, (err, row) => {
        if (err) {
          res.status(400).json({"error":err.message})
          return
        }
        var topic = row.topic
        var quiz = test.parseTest(row.test)
        res.json({
            "message":"Операція успішна",
            "data": {topic},
            quiz
        })
    })
})

// Add an Material
router.post("/api/materials", (req, res, next) => {
    if (auth.isAdmin(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var data = {
        course: req.body.course,
        topic: req.body.topic,
        comment: req.body.comment,
        pdf: req.files?.pdf?.data || null,
        test: req.body.test,
        date: new Date().toISOString().replace('T', ' ').replace('Z', '')
    }
    var sql = "INSERT INTO materials (course, topic, comment, pdf, test, date) VALUES
(?,?,?,?,?,?)"
    var params = [data.course, data.topic, data.comment, data.pdf, data.test, data.date]

    db.run(sql, params, function (err, result) {
        if (err) {
            res.status(400).json({"error": err.message})
            return
        }
        res.json({
            "message": "Операція успішна",
            "data": data,
            "id" : this.lastID
        })
    })
})

// Edit an Material by Id
router.patch("/api/materials/:id", (req, res, next) => {
    if (auth.isAdmin(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var data = {
        course: req.body.course,
        topic: req.body.topic,
        comment: req.body.comment,
        pdf: req.files?.pdf?.data || null,
        test: req.body.test,
        date: new Date().toISOString().replace('T', ' ').replace('Z', '')
    }
```

```
    var sql = `UPDATE materials set
    course = COALESCE(?,course),
    topic = COALESCE(?,topic),
    comment = COALESCE(?,comment),
    pdf = COALESCE(?,pdf),
    test = COALESCE(?,test),
    date = COALESCE(?,date)
    WHERE id = ?`
    var params = [data.course, data.topic, data.comment, data.pdf, data.test, data.date,
req.params.id]

    db.run(sql, params,
        function (err, result) {
            if (err){
                res.status(400).json({"error": res.message})
                return
            }
            res.json({
                message: "Операція успішна",
                data: data,
                changes: this.changes
            })
    })
})

// Delete an Material by id
router.delete("/api/materials/:id", (req, res, next) => {
    if (auth.isAdmin(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var sql = "DELETE FROM materials WHERE id = ?"
    var params = [req.params.id]

    db.run(sql, params, function (err, Certificate) {
        if (err) {
            res.status(400).json({"error": res.message})
            return
        }
        res.json({"message":"Операція успішна", changes: this.changes})
    })
})

module.exports = router

// CoursesComponent.js

<template>
    <section class="bg-light" v-if="this.$store.state.isUserAuth == true ||
this.$store.state.isAdminAuth == true">
        <div class="container p-3">
            <div class="row justify-content-center align-items-center">
                <div class="col">
                    <div class="text-center m-3">
                        <h1 class="text-dark">Курси</h1>
                        <p class="text-black-50">перелік курсів</p>
                    </div>
                </div>
                <div class="col">
                    <div class="card m-3" v-for="(item, index) in
data.data" :key="index">
                        <div class="card-body">
```

```
                            <h5 class="card-title"><strong>{{ item.
name }}</strong></h5>
                            <p class="card-text">{{ item. comment }}</p>
                            <router-link :to="'/materials/' + item.id" class="btn btn-
primary w-100" data-bss-hover-animate="pulse">Перейти до матеріалів курсу</router-link>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>
    <section class="bg-light" v-else>
        <div class="container p-3">
            <div class="alert alert-primary m-3" role="alert">
                Будь ласка, авторизуйтесь, щоб побачити контент.
            </div>
        </div>
    </section>
</template>

<script>
export default
{
    name: 'CoursesComponent',
    data: function()
    {
        return {
            url: 'api/courses/',
            data: []
        }
    },
    methods:
    {
        getData: async function()
        {
            await this.$api.get(this.url)
            .then((response) =>
            {
                this.data = response.data;
            })
            .catch((error) =>
            {
                this.status = error.response.data.message;
            });
        },
    },
    mounted()
    {
        this.getData();
    },
    unmounted()
    {

    }
}
</script>


// MaterialsComponent.js

<template>
    <section class="bg-light">
```

```
        <div class="container p-3">
            <div class="row row-cols-1 justify-content-center align-items-center">
                <div class="col">
                    <div class="text-center m-3">
                        <h1 class="text-dark">Матеріали до теми</h1>
                        <p class="text-black-50">перелік матеріалів до теми</p>
                    </div>
                </div>
                <div class="col">
                    <div class="card m-3" v-for="(item, index) in
data.data" :key="index">
                        <div class="card-body">
                            <h5 class="card-
title"><strong>{{ item.topic }}</strong></h5>
                            <p class="card-text">{{ item. comment }}</p>
                            <div class="row row-cols-1 justify-content-center align-
items-center">
                                <div class="col">
                                    <button class="btn btn-primary w-100 m-1"
@click="pdfConverter(item.pdf)" data-bss-hover-animate="pulse">Навчальний
матеріал</button>
                                </div>
                                <div class="col">
                                    <router-link :to="'/test/' + item.id" class="btn
btn-primary w-100 m-1" data-bss-hover-animate="pulse">Тест</router-link>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>
</template>

<script>
export default
{
    name: 'MaterialsComponent',
    data: function()
    {
        return {
            url: 'api/materials/',
            data: []
        }
    },
    methods:
    {
        pdfConverter: function(pdfData)
        {
            const byteArray = new Uint8Array(pdfData.data);
            const blob = new Blob([byteArray], { type: 'application/pdf' });
            const pdfUrl = URL.createObjectURL(blob);
            window.open(pdfUrl);
        },
        getData: async function()
        {
            await this.$api.get(this.url + this.$route.params.id)
            .then((response) =>
            {
                this.data = response.data;
            })
```

```
            .catch((error) =>
            {
                this.status = error.response.data.message;
            });
        },
    },
    mounted()
    {
        this.getData();
    },
    unmounted()
    {

    }
}
</script>

// CoursesView.js

<template>
<NavBarComponent/>

<CoursesComponent/>

<FooterComponent/>
</template>

<script>
import NavBarComponent from '@/components/template/NavBarComponent.vue'
import CoursesComponent from '@/components/courses/CoursesComponent.vue'
import FooterComponent from '@/components/template/FooterComponent.vue'

export default
{
    name: 'CoursesView',
    components:
    {
        NavBarComponent,
        CoursesComponent,
        FooterComponent
    },
}
</script>

// MaterialsView.js

<template>
<NavBarComponent/>

<MaterialsComponent/>

<FooterComponent/>
</template>

<script>
import NavBarComponent from '@/components/template/NavBarComponent.vue'
import MaterialsComponent from '@/components/materials/MaterialsComponent.vue'
import FooterComponent from '@/components/template/FooterComponent.vue'

export default
{
    name: 'MaterialsView',
```

```
    components:
    {
        NavBarComponent,
        MaterialsComponent,
        FooterComponent
    },
}
</script>
```

ДОДАТОК Б. Слайди мультимедійної презентації

Дипломна робота

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

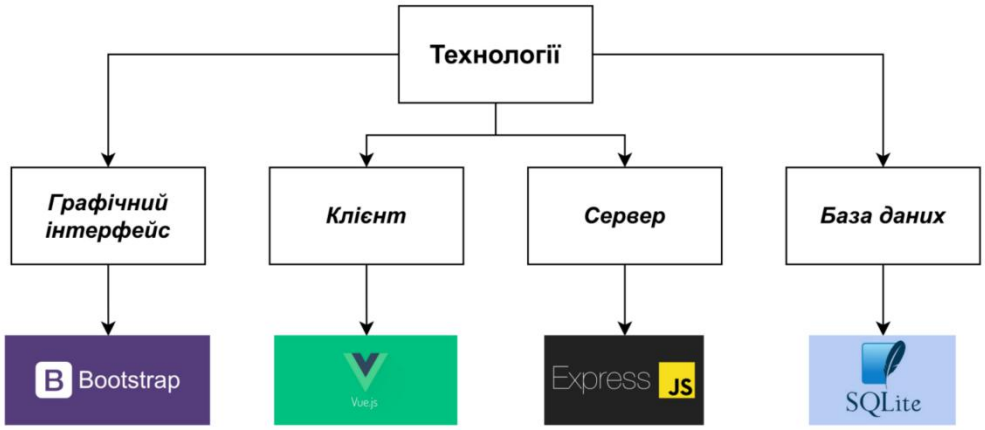
Розробка універсальної веб-орієнтованої платформи з онлайн курсів для навчання та перевірки знань

Гаврилюк Владислав Сергійович

Основні відомості

- Онлайн-навчання, що активно розвивається завдяки цифровим технологіям, потребує ефективних веб-платформ для організації освітнього процесу, зокрема інтерактивних курсів, систем управління матеріалами та інструментів оцінювання знань.
- Метою даної роботи є створення універсальної веб-платформи для онлайн-курсів, що оптимізує освітній процес за допомогою передових технологій, таких як SQL та Node.js.
- Практичне значення розробки полягає в забезпеченні високого рівня обслуговування освітніх установ та студентів, підвищуючи якість освіти і конкурентоспроможність.

Стек технологій



UX-мапа застосунку

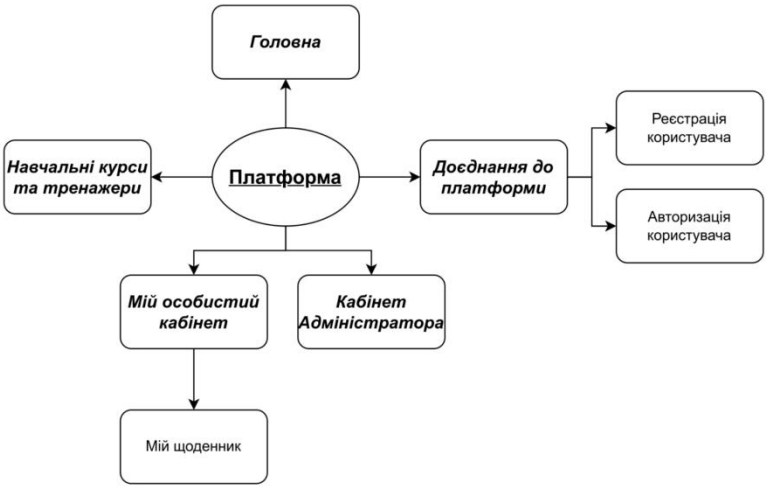
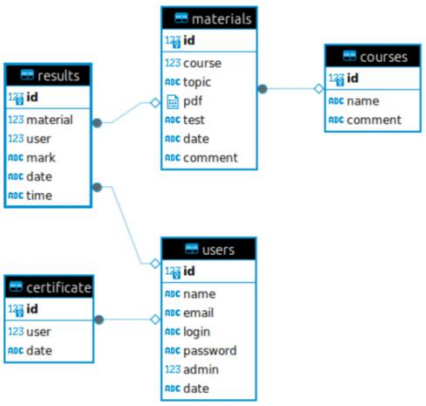
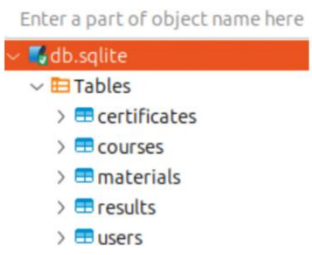
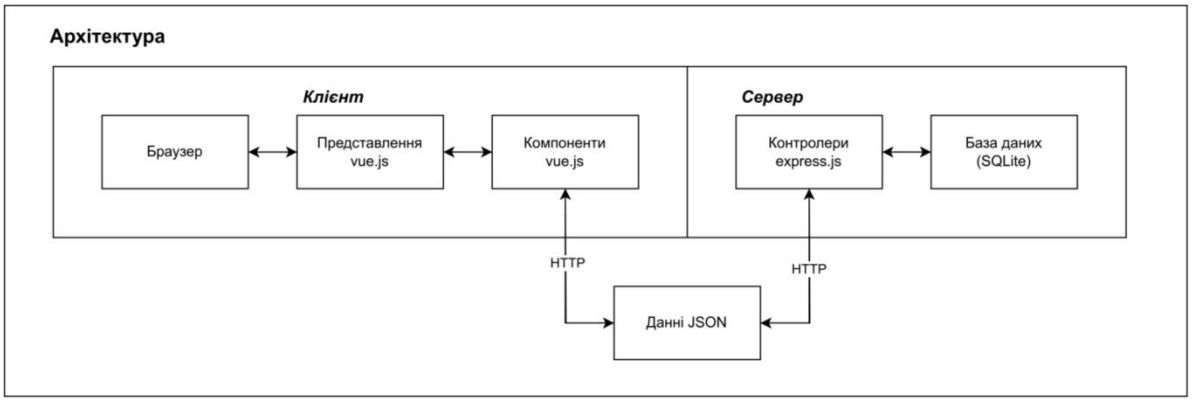


Схема БД





Клієнт-серверна архітектура



Функціональність системи

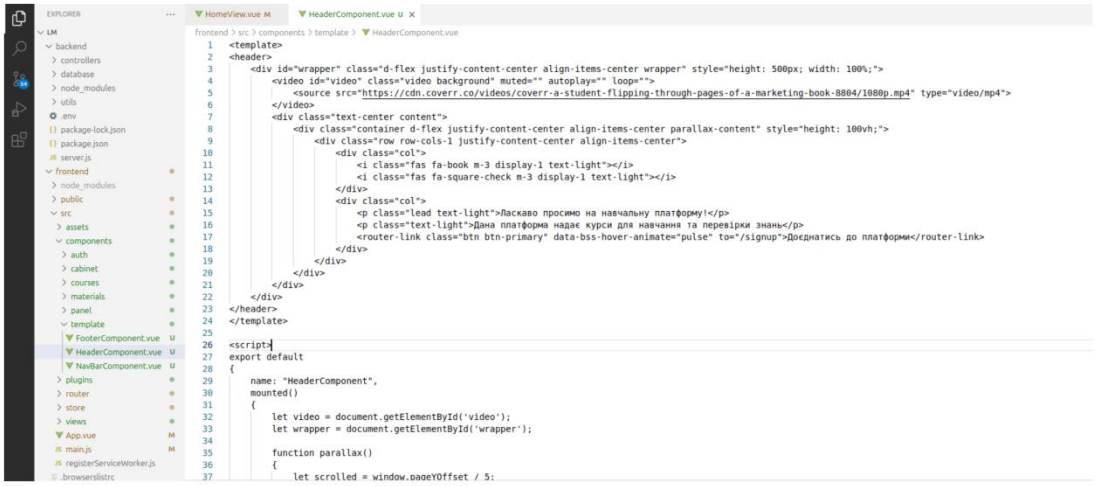
Адміністратор:

- Авторизація
- Створення курсів
- Додавання матеріалів
- Перегляд успішності

Користувач:

- Реєстрація
- Авторизація
- Перегляд курсів
- Прохолження матеріалів
- Перегляд успішності

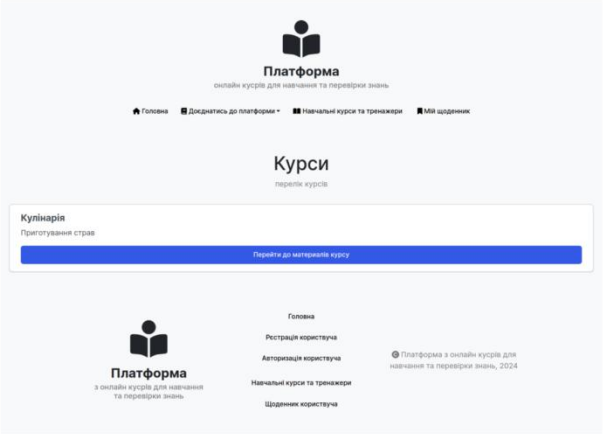
Процес розробки

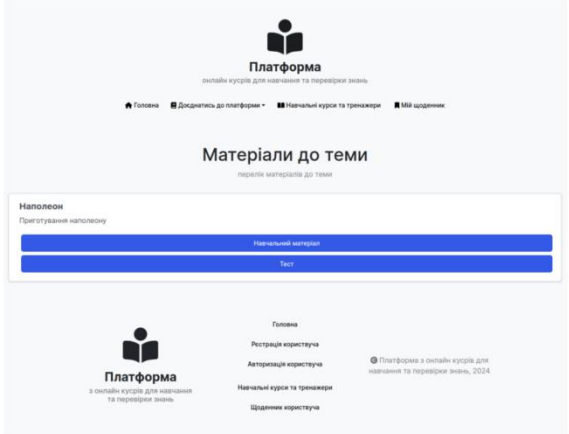


Метод на отримання матеріалів курсу

```
      // Get a List of Materials by Course
router.get("/api/materials/:id", (req, res, next) => {
    if (auth.isUser(req.headers.authorization) == false) {
        return res.status(401).json({ message: "Не авторизовано" })
    }

    var sql = `SELECT * FROM materials WHERE course = ?`
    var params = [req.params.id]

    db.all(sql, params, (err, rows) => {
        if (err) {
            res.status(400).json({ "error": err.message })
            return
        }

        res.json({
            "message": "Операція успішна",
            "data": rows
        })
    })
})
```

Сторінки з курсами та матеріалами до них





Дипломна робота

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Дякую за увагу!

Гаврилюк Владислав Сергійович

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Гаврилюка Владислва Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність: *121 «Інженерія програмного забезпечення»*

Освітня програма: *«Розробка програмного забезпечення»*

Тема дипломного проекту: *Розробка універсальної веб-орієнтованої платформи з онлайн курсів для навчання та перевірки знань*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 81 сторінку. У пояснювальній записці описано етапи розробки універсальної веб-орієнтованої платформи з онлайн курсів для навчання та перевірки знань засобами PHP та MySQL. Графічна частина складається з окремих слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Гаврилюк Владислав поступово та послідовно виконував всі етапи, проявляв ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Гаврилюк Владислав під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Гаврилюк Владислав показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування та математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, оформлювати слайди та складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як MS VS Code, PHP, MySQL, MS PowerPoint, MS Visio та ін.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Жадан Артур Сергійович*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерної техніки та програмної інженерії*

Підпис

«10» 06 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Гаврилюка Владислава Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

Керівник дипломного проекту (роботи) *Жадан Артур Сергійович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка універсальної веб-орієнтованої платформи з онлайн курсів для навчання та перевірки знань*

Обсяг розрахунково-пояснювальної записки *81* сторінок

Обсяг графічної (презентаційної) частини *11* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі навчання та перевірки знань та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 11 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скриншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Реалізовано універсальну веб-орієнтовану платформу з онлайн курсів для навчання та перевірки знань, що дозволяє створювати курси, додавати до них матеріали та проходити тести.*

Веб-застосунок створено засобами сучасним технологій – vue.js та express.js.

Колірна гама графічного інтерфейсу відповідає предметній області.

д) основні недоліки дипломного проекту

1. *Недостатній опис структури платформи та етапів проектування.*
2. *Для серверної частини веб-застосунку використовувалась процедурна парадигма програмування замість об'єктно-орієнтованої, але у роботі не представлено блок-схем алгоритмів.*
3. *Наявні помилки оформлення тексту пояснювальної записки*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *Царьов Роман Юрійович*

Місце роботи і посада рецензента *Державний університет інтелектуальних технологій і зв'язку, ст. викладач, зав. кафедри комп'ютерної інженерії та інформаційних систем*

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ СУІТЗ
18.06.2024 р.

ВІДДІЛ КАДРІВ

Підпис:

«18» 06 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016336074

Дата перевірки:
08.06.2024 19:56:56 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2024 20:01:52 EEST

ID користувача:
100011688

Назва документа: 4РП-07_Гаврилюк_В

Кількість сторінок: 47 Кількість слів: 6814 Кількість символів: 52886 Розмір файлу: 2.95 MB ID файлу: 1016136833

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

7.59%
Схожість

Найбільша схожість: 1.42% з Інтернет-джерелом (https://developerhowto.com/2018/12/29/build-a-rest-api-with-node-js-..

7.59% Джерела з Інтернету | 622 ..... Сторінка 49

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування | 12 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Гаврилюк Владислав Сергійович,
здобувач освіти гр. 4РП-07, та

Жадан Артур Сергійович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка універсальної веб-орієнтованої платформи з онлайн курсів для навчання та перевірки знань» (автор роботи – Гаврилюк В.С., керівник роботи – Жадан А.С.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Гаврилюк В.С. /

Керівник _____ / Жадан А.С. /

«10» червня 2024 р.