

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет
Кафедра комп'ютерно-інтегрованих оптичних та навігаційних
систем**

Затверджую:

Завідувач кафедри

_____ Надія БУРАУ

« ____ » _____ 20 __ р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Комп'ютерно - інтегровані технології та
системи навігації і керування»**

спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

на тему: Платформа для онлайн-навчання

Виконав:

студент 4 курсу, групи ПГ-91

Гребенюк Андрій Володимирович

Керівник:

Асистент, к.т.н. Рупіч С. С.

Рецензент:

Доцент, к.т.н. Шевченко В. В.

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____

**Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»**

Інститут (факультет) _____ Приладобудівний _____
(повна назва)

Кафедра _____ Комп'ютерно-інтегрованих оптичних та навігаційних систем _____
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність _____ 151 – Автоматизація та комп'ютерно-інтегровані технології _____
(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Н.І. Бурау _____
(підпис) (ініціали, прізвище)

« _____ » _____ 20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Гребенюку Андрію Володимировичу _____

(прізвище, ім'я, по батькові)

1. Тема дипломної роботи: Платформа для онлайн-навчання

Науковий керівник: _____ Рупіч Сергій Сергійович, к.т.н. _____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «30» _____ травня 2023р. № 2057-с

2. Термін подання студентом роботи «05» червня 2023 р.

3. Вихідні дані до роботи: реалізувати систему у вигляді десктопного або веб-додатку, реалізувати можливість керування курсами, реалізувати можливість завантажувати навчальні матеріали та літературу та надання доступу до них.

4. Питання, що мають бути розроблені:

1. Зробити огляд та аналіз актуальності проблеми.
2. Визначити вимоги до застосунку.
3. Побудувати структуру проекту.
4. Описати функціональність та особливості системи.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): таблиці, графіки, рисунки тощо.

6. Дата видачі завдання 17.04.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд стану проблеми	23.04.2023	
2.	Визначення вимог до застосунку	26.04.2023	
3.	Побудова структури проекту	30.04.2023	
4.	Розробка застосунку	25.05.2023	
5.	Тестування готового програмного продукту та виправлення помилок	31.05.2023	
6.	Оформлення пояснювальної записки. Підготовка до захисту	05.06.2023	

Студент

(підпис)

А. В. Гребенюк
(ініціали, прізвище)

Керівник дипломної роботи

(підпис)

С. С. Рупіч
(ініціали, прізвище)

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломної роботи складається з двох розділів, містить 20 рисунків та 31 джерело.

Мета. Дипломна робота присвячена дослідженню актуальності платформ онлайн навчання та розробці веб-інтерфейсу для забезпечення якісного та зручного процесу навчання в онлайн-середовищі. У сучасному світі, де швидкість розвитку технологій надзвичайно висока, платформи онлайн навчання набули великого значення, особливо у зв'язку з пандемією COVID-19, яка спричинила значний зріст популярності дистанційного навчання.

У роботі проводиться аналіз різних платформ онлайн навчання, їхніх переваг та недоліків. Досліджуються основні функціональні можливості, інтерфейс та зручність використання цих платформ для студентів і викладачів. Звертається увага на важливі аспекти, такі як надійність, безпека даних та можливості інтерактивної взаємодії.

На основі проведеного аналізу розробляється веб-інтерфейс для онлайн навчання, який враховує зібрані дані. Цей проєкт має на меті покращити досвід навчання в онлайн-середовищі шляхом забезпечення простого та ефективного способу доступу до навчального матеріалу, зручного спілкування між студентами та викладачами, а також моніторингу та оцінювання навчальних досягнень.

Отримані результати дослідження та розробки веб-інтерфейсу можуть бути корисними для університетів, навчальних закладів та всіх зацікавлених учасників освітнього процесу. Результати роботи вказують на те, що платформи онлайн навчання є необхідними та важливими інструментами для розвитку освіти в епоху цифрових технологій.

ABSTRACT

Structure and scope of the work. The explanatory note of the thesis consists of two chapters, contains 20 figures and 31 sources.

Goal. The thesis is devoted to the study of the relevance of online learning platforms and the development of a web application to ensure a high-quality and convenient learning process in the online environment. In today's world, where the speed of technology development is extremely high, online learning platforms have become very important, especially in connection with the COVID-19 pandemic, which has led to a significant increase in the popularity of distance learning.

This paper analyzes various online learning platforms, their advantages and disadvantages. The main functionalities, interface and usability of these platforms for students and teachers are investigated. Attention is drawn to important aspects such as reliability, data security, and interactive interaction.

Based on the analysis, a web interface for online learning is being developed that takes into account the collected data. This project aims to improve the learning experience in an online environment by providing a simple and effective way to access learning material, convenient communication between students and teachers, and monitoring and evaluation of learning achievements.

The obtained results of the research and development of the web interface can be useful for universities, educational institutions and all interested participants in the educational process. The results of the work indicate that online learning platforms are necessary and important tools for the development of education in the digital age.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ.....	7
ВСТУП.....	9
РОЗДІЛ 1. ЗАДАЧА СТВОРЕННЯ ПЛАТФОРМИ ДЛЯ ОНЛАЙН НАВЧАННЯ .	11
1.1. Бази даних для платформ онлайн навчання	12
1.1.1. Реляційні бази даних.....	12
1.1.2. NoSQL бази даних.....	13
1.1.3. Графові бази даних.....	13
1.1.4. Застосування баз даних у платформах онлайн-навчання	14
1.2. Проектування систем управління для платформ онлайн-навчання.....	15
1.3. Питання безпеки та конфіденційності в платформах онлайн навчання	17
1.4. Аналіз інтерфейсу користувача для платформи онлайн навчання	18
1.5. Методи оцінювання та контролю знань у віртуальному навчальному середовищі	19
1.6. Проблеми платформ онлайн навчання та порівняння їх між собою	21
1.7. Огляд аналогів платформ для онлайн навчання	22
РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ	26
2.1. Вибір інструментів для розробки	27
2.2. Функціональність та особливості MongoDB онлайн-платформи.	31
2.3. Опис інтерфейсу та його функціонал.	35
ВИСНОВКИ.....	44
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	46
Додаток А.....	50

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

CSS (Cascading Style Sheets) - це мова стилів, що використовується для оформлення і форматування веб-сторінок. Вона визначає зовнішній вигляд елементів на сторінці, таких як кольори, шрифти, розміри, розташування, фонові зображення та багато іншого.

NoSQL (Not Only SQL) - це підхід до зберігання та обробки даних, який відрізняється від традиційної реляційної моделі даних, що використовує SQL. NoSQL бази даних призначені для роботи з великими обсягами неупорядкованих та незв'язаних даних, які можуть бути структуровані або напівструктуровані.

SQL (Structured Query Language) - це мова запитів, яка використовується для управління та маніпуляції даними у реляційних базах даних. SQL надає стандартизований спосіб виконання операцій, таких як створення, модифікація та видалення таблиць та записів, а також виконання запитів для отримання необхідної інформації.

ID - це скорочення від "ідентифікатор". ID використовується для унікальної ідентифікації об'єкта або запису в системі. Це числовий або текстовий код, який відповідає конкретному елементу в базі даних, документі, таблиці або іншій структурі даних.

JSON (JavaScript Object Notation) - це формат обміну даними, який використовується для зберігання та передачі структурованих інформаційних об'єктів.

BSON (Binary JSON) - це бінарний формат обміну даними, який використовується для зберігання та передачі структурованих об'єктів даних. BSON є розширенням формату JSON і дозволяє більш ефективну серіалізацію та десеріалізацію даних.

LMS (Learning Management System) - це система управління навчанням, що використовується для організації та поширення освітнього матеріалу в електронному форматі

HTML (HyperText Markup Language) - це мова розмітки, що використовується для створення та організації веб-сторінок.

GUI (Graphical User Interface) - це графічний інтерфейс користувача, що надає зручний спосіб взаємодії між користувачем та комп'ютерною системою.

ВСТУП

Сучасний світ в постійному процесі трансформації вирішення нових викликів, що виникають, і разом з ним розвиваються інноваційні технології. Одним з проявів цього є значне збільшення популярності онлайн-навчання. За останні роки онлайн-навчання стало не просто альтернативою традиційній формі отримання знань, але і важливим інструментом, що задовольняє потреби сучасного суспільства [1].

По-перше, причина переходу в онлайн-навчання пов'язана з розвитком технологій і доступністю мережі Інтернет. Сучасні інформаційні технології надають можливість легко отримувати доступ до різноманітних джерел інформації з будь-якого куточка світу. Інтернет став основним засобом комунікації, спілкування та обміну знаннями. Тому люди все частіше вибирають онлайн-формат навчання, оскільки він зручний та гнучкий.

По-друге, онлайн-навчання дозволяє отримати освіту на високому рівні незалежно від місця проживання. Це особливо актуально для людей, які мешкають у віддалених районах або маленьких селах, де немає можливості отримати якісну освіту. Завдяки онлайн-навчанню, їм відкривається доступ до курсів із будь-якої галузі знань, які проводять визнані експерти. Така можливість дає змогу розвиватися та здобувати нові навички незалежно від географічних обмежень.

По-третє, зростає потреба у неперервному самовдосконаленні та професійному розвитку. Сучасний світ змінюється дуже швидко, і для успішної кар'єри необхідно постійно оновлювати свої знання та навички. Онлайн-навчання надає можливість вчитися у зручний для себе час та темп, а також обирати саме ті курси, які цікавлять та корисні для подальшого розвитку. Крім того, онлайн-курси часто мають інтерактивний формат, що дозволяє студентам активно взаємодіяти з викладачами та іншими учасниками курсу, обмінюючись думками та досвідом [2].

Варто зазначити про економічну складову онлайн-навчання. Часто воно є доступнішим з фінансової точки зору порівняно з традиційною освітою. Відсутність необхідності витрачати кошти на дорого-вартісні підручники та

додаткові матеріали, а також можливість отримати навчання від провідних експертів дистанційно робить онлайн-навчання привабливим з практичної точки зору.

Загалом, онлайн-навчання має багато переваг, які пояснюють його популярність. Зручність, доступність, гнучкість та можливість навчатися на високому рівні без обмежень за місцем проживання - ось ті фактори, які зробили онлайн-навчання актуальним і популярним на сучасному етапі. Тому не випадково все більше людей обирають цей спосіб отримання знань та розвитку. Використання онлайн-навчання стає необхідною складовою успішної освіти у XXI столітті. Завдання дипломної роботи є створення онлайн платформи для навчання.

РОЗДІЛ 1. ЗАДАЧА СТВОРЕННЯ ПЛАТФОРМИ ДЛЯ ОНЛАЙН НАВЧАННЯ

Створення системи для онлайн курсів - це складний і багатокроковий процес, що вимагає глибокого розуміння педагогічних принципів, технологій електронного навчання та програмування. Для розробки такої системи необхідно врахувати кілька ключових етапів, таких як планування, проектування, розробка, впровадження та оцінка. У цьому розділі будуть розглянуті основні кроки та складові створення системи для онлайн курсів на основі наукової літератури та реальних веб-джерел.

Етап планування є першим кроком у розробці системи для онлайн курсів [3]. На цьому етапі необхідно визначити мету та цілі курсу, а також цільову аудиторію. Дослідження потреб учнів та аналіз конкурентних систем є важливою частиною планування. Також потрібно розглянути технологічні та бюджетні обмеження для визначення обсягу робіт та ресурсів, необхідних для розробки системи.

Після планування настає етап проектування, на якому визначаються основні архітектурні принципи та функціональні можливості системи [4]. Розробка структури курсу, його модулів та лекційних матеріалів є важливою частиною проектування. Потрібно врахувати наукові методи навчання, інтерактивність, персоналізацію та можливості зворотного зв'язку для ефективного навчання учнів.

Після проектування настає етап розробки системи, який включає програмування, створення веб-інтерфейсу та інтеграцію з необхідними засобами. Програмування вимагає використання різних мов програмування, таких як HTML, CSS, JavaScript та інші, для створення функціональності системи. Розробка веб-інтерфейсу повинна бути зручною та інтуїтивно зрозумілою для користувачів. Інтеграція засобів, таких як системи управління навчанням (LMS), системи відеоконференцій та засоби оцінювання, дозволить покращити функціональність системи та забезпечити її повноту.

Після розробки настає етап впровадження системи для онлайн курсів. На цьому етапі необхідно провести пілотне тестування системи та забезпечити навчання для обмеженої аудиторії [5]. Важливо врахувати зворотний зв'язок від учасників курсу для виявлення можливих проблем та вдосконалення системи. Після успішного впровадження системи можна перейти до остаточного етапу - оцінки та звітування про результати системи для забезпечення її ефективності.

Платформа для онлайн-навчання є актуальною та широко використовуваною технологією в сучасному освітньому середовищі.

1.1. Бази даних для платформ онлайн навчання

Одним з ключових елементів платформ для онлайн-навчання є бази даних. Вони використовуються для зберігання і організації різноманітної інформації, такої як дані користувачів, курси, матеріали, завдання та інші компоненти, необхідні для проведення ефективного навчання. Далі буде розглянуто різні типи баз даних, включаючи реляційні, NoSQL та графові бази даних, а також їхнє застосування в контексті платформ для онлайн-навчання.

1.1.1. Реляційні бази даних

Реляційні бази даних є одним з найпоширеніших типів баз даних і використовуються в багатьох платформах для онлайн-навчання. Вони ґрунтуються на реляційній моделі даних, де дані зберігаються у вигляді таблиць зі стовпцями і рядками. Кожна таблиця представляє сутність або відношення, а стовпці визначають атрибути цієї сутності. Зв'язки між таблицями встановлюються за допомогою ключів, які пов'язують відповідні записи.

Реляційні бази надають структурований підхід до зберігання даних, що дозволяє ефективно виконувати запити і забезпечувати їх цілісність. У контексті платформ для онлайн-навчання, реляційні бази даних можуть використовуватись для зберігання інформації про користувачів, такі як особисті дані, дані авторизації та інші атрибути. Вони також можуть слугувати для зберігання даних про курси, включаючи інформацію про вчителів, студентів, матеріали курсів та результати оцінювання [6].

1.1.2. NoSQL бази даних

Поряд з реляційними базами даних, для онлайн-навчання також можуть використовуватися такі бази як NoSQL (Not only SQL). NoSQL є альтернативою реляційним системам і надають більш гнучкий підхід до зберігання та обробки даних. Вони особливо корисні в ситуаціях, де потрібно обробляти великі обсяги даних з високою швидкістю [7].

Ці бази даних можуть бути документ-орієнтованими або графовими. Вони надають можливість гнучко зберігати інформацію різної структури і не потребують жорсткого визначення її схеми. Це особливо корисно для платформ онлайн-навчання, де можуть бути різноманітні матеріали, наприклад: відео, аудіо, текстові матеріали та інші, які можуть бути збережені у відповідних форматах [8].

1.1.3. Графові бази даних

Графові бази даних [9] засновані на теорії графів і використовуються для моделювання зв'язків між об'єктами. Вони складаються з вузлів, які представляють

об'єкти, і ребер, які показують зв'язки між цими об'єктами. Кожен вузол і ребро можуть мати властивості, які дозволяють додатково описувати ці об'єкти та їхні зв'язки.

Графові бази підходять для моделювання складних залежностей та аналізу зв'язків між елементами. У платформах для онлайн-навчання, ці бази можуть бути використані для моделювання взаємностей між користувачами, їхніми інтересами, спільними курсами та іншими взаємодіями.

1.1.4. Застосування баз даних у платформах онлайн-навчання

Враховуючи вище зазначене, можна виділити, що бази даних відіграють ключову роль у платформах для онлайн-навчання, забезпечуючи зберігання, доступ та організацію інформації, необхідної для проведення навчання. Вони надають можливість зберігати дані користувачів, їхні особисті дані, історію навчання, результати оцінювання, матеріали курсів та іншу важливу інформацію.

У платформах онлайн-навчання часто використовують комбінацію різних типів баз даних в залежності від потреб конкретної системи.

Використання відповідного інструменту залежить від вимог системи, типу інформації та потреб у обробці та аналізі. Знання про різні типи баз даних допоможе студентам ефективно впроваджувати та управляти ними у своїх проектах з платформами онлайн-навчання.

У цій роботі було використано нереляційну базу даних MongoDB. MongoDB - це нереляційна база даних, яка використовує документи у форматі JSON для зберігання і організації даних [10]. Вона відноситься до категорії NoSQL, що означає, що вона не використовує традиційні реляційні таблиці зі структурованими схемами. MongoDB надає можливість зберігати дані у вигляді документів, які представлені у форматі BSON (Binary JSON). BSON є бінарним представленням

JSON-подібних структур інформації, що дозволяє більш ефективно зберігати та обробляти її [11]. Основною одиницею даних в MongoDB є документ, який може містити будь-яку кількість полів з різними типами даних. Документи групуються в колекції, які відповідають реляційним таблицям у традиційних програмах. Кожен документ має унікальний ідентифікатор (`_id`), що дозволяє однозначно ідентифікувати його в межах колекції.

Загалом, MongoDB є потужним інструментом, який дозволяє зберігати, організовувати та обробляти дані у гнучкому форматі, що робить його популярним вибором для багатьох сучасних додатків та систем.

1.2. Проектування систем управління для платформ онлайн-навчання

Ефективне управління платформою для онлайн-навчання вимагає розробки відповідної системи управління. Далі буде розглянуто різні підходи до проектування систем управління, включаючи централізовані та децентралізовані моделі. Починаючи з опису основних функціональних вимог до систем управління, таких як автентифікація та авторизація користувачів, управління курсами та матеріалами, оцінювання та звітність, а також забезпечення безпеки та конфіденційності і ключові аспекти цих підходів.

1. Централізована модель управління [12]. У цій моделі система управління розташована на центральному сервері, який контролює всі аспекти платформи для онлайн-навчання. Централізована модель надає великий контроль над системою, дозволяє одночасно керувати курсами, користувачами, оцінками та іншими аспектами платформи. Однак, цей підхід може мати деякі обмеження, наприклад якщо багато користувачів одночасно взаємодіють з системою, централізований сервер може стати перевантаженим, що впливає на швидкість відповіді та затримки при обробці запитів. Також, оскільки централізована модель базується на

одному сервері, вона стає потенційною точкою вразливості. Якщо сервер відмовить або він буде не захищений від кібератак, це може призвести до зупинки всієї системи.

2. Децентралізована модель управління [12]. У цій моделі функції управління розподілені між різними компонентами платформи. Наприклад, можуть бути використані розподілені бази даних або розподілені сервери для управління користувачами, курсами та іншими аспектами системи. Цей підхід може забезпечити більшу масштабованість та ефективність системи, особливо у випадку великої кількості користувачів.

До основних функціональних вимог до систем управління платформою онлайн-навчання включають [13]:

- Автентифікація та авторизація користувачів. Система повинна забезпечувати ідентифікацію користувачів та контроль доступу до різних ресурсів платформи. Це може включати в себе механізми автентифікації, створення облікових записів користувачів, ролей та прав доступу.
- Управління курсами та матеріалами. Система повинна надавати зручні інструменти для створення, редагування та управління курсами, що включає в себе: структурування матеріалів, завдань, тестів, домашніх завдань та іншого змісту, який використовується в процесі навчання.
- Оцінювання та звітність. Система повинна надавати можливість робити оцінювання навчального прогресу студентів, збирати результати тестів та завдань, а також надавати звіти вчителям та адміністраторам. Можливість моніторингу прогресу та аналізу даних може бути корисною для вдосконалення процесу навчання.
- Забезпечення безпеки та конфіденційності. Система повинна мати механізми захисту даних користувачів, включаючи персональну інформацію та результати навчання. Для забезпечення безпеки та

конфіденційності даних можуть використовуватись криптографічні протоколи, механізми контролю доступу та інші технічні засоби

1.3. Питання безпеки та конфіденційності в платформах онлайн навчання

Питання безпеки та конфіденційності в платформах онлайн навчання є важливими аспектами, які потребують ретельного вивчення та впровадження відповідних заходів для забезпечення захисту інформації користувачів. З огляду на зростаючу популярність електронного навчання, виникає необхідність у розумінні та вирішенні цих питань для забезпечення безпеки та конфіденційності учасників навчального процесу.

Один з найважливіших аспектів безпеки в платформах онлайн навчання - це захист особистих даних користувачів. Під час реєстрації на платформі, учасники надають свої персональні дані, такі як ім'я, прізвище, адреса електронної пошти тощо. Ці дані потребують особливої уваги з точки зору конфіденційності, оскільки їх може бути використано зловмисниками для незаконних цілей, таких як крадіжка особистої ідентифікації або шахрайство. Тому важливо розробляти та впроваджувати механізми шифрування та захисту цих даних, а також встановлювати політику конфіденційності, що регламентує обробку та зберігання особистої інформації.

Крім того, у платформах онлайн навчання виникають питання щодо безпеки під час передачі даних між користувачем та сервером.. З метою забезпечення конфіденційності даних під час їх передачі використовуються шифрувальні протоколи та методи аутентифікації, такі як протоколи SSL/TLS. Це дозволяє захистити інформацію від небажаного доступу та зберегти її цілісність.

Також стоїть завдання забезпечення безпеки під час використання різноманітних інтерактивних елементів платформи. Наприклад, учасники можуть взаємодіяти між собою через чат або форуми, де важливо контролювати зміст

повідомлень та уникати поширення шкідливих вірусів або спаму. Також, під час проведення тестів або виконання завдань, потрібно забезпечити надійність та недоступність шахрайських дій, таких як плагіат або підробка результатів.

Для ефективного забезпечення безпеки та конфіденційності в платформах онлайн навчання необхідно розробляти і впроваджувати комплексні системи захисту, які охоплюють як технічні, так і організаційні аспекти. Важливо проводити систематичні аудити безпеки, виявляти потенційні загрози та розробляти стратегії протидії ним. Також, важливо враховувати законодавчі вимоги щодо захисту персональних даних та дотримуватись принципів етичного використання інформації [14].

1.4. Аналіз інтерфейсу користувача для платформи онлайн навчання

Інтерфейс платформи для онлайн-навчання є одним з найважливіших елементів, що впливає на користувачів і сприяє їх задоволенню від процесу навчання.

Один з популярних підходів до реалізації інтерфейсу в платформах онлайн-навчання - це простота та зручність використання. Він повинен бути інтуїтивно зрозумілим, добре структурованим та навіть незважаючи на велику кількість функціональності, повинен залишатися практичним у використанні. Чіткі та зрозумілі іконки, меню та розділи надають можливість користувачам швидко орієнтуватися в платформі [15].

Інший метод, який ефективно використовують, - це персоналізація інтерфейсу. Здатність адаптуватися до потреб користувача і надавати індивідуальний досвід навчання - це важливий аспект платформи. Наприклад, можливість робити особисті налаштування, обирати предмети або теми навчання, вибирати формати матеріалів, які найкраще відповідають їхнім потребам [16].

У платформах онлайн-навчання також активно використовуються методи, спрямовані на залучення і мотивацію користувачів. Наприклад, системи нагород та досягнень, які стимулюють їх досягати поставлених цілей і здобувати знання. Використання ігрових елементів або інтерактивних завдань також робить процес навчання більш цікавим та захоплюючим.

Інтерфейс платформи повинен бути також спрямований на підтримку комунікації між користувачами. Форуми, чати або можливість обговорення матеріалів навчання з іншими учасниками навчання сприяють активному обміну знаннями, спільним проектам та співпраці.

Враховуючи особливості онлайн-навчання, веб-сторінка повинна бути добре організована, для забезпечення зручного пошуку, сортування та навігацію по матеріалах навчання. Важливо також забезпечити зручний доступ до домашніх завдань, оцінок та зворотного зв'язку з викладачами [17].

1.5. Методи оцінювання та контролю знань у віртуальному навчальному середовищі

Методи оцінювання та контролю знань у віртуальному навчальному середовищі є ключовими елементами успішного проведення онлайн навчання. Завдяки цим методам, викладачі можуть визначати рівень розуміння матеріалу студентами, оцінювати їх навчальний прогрес та надавати зворотний зв'язок для подальшого вдосконалення навчання. Далі буде розглянуто детальніше декілька ефективних методів оцінювання та контролю знань у віртуальному навчальному середовищі.

Один з найпоширеніших методів оцінювання в онлайн навчанні - тестування. Цей метод дозволяє викладачам оцінити рівень засвоєння матеріалу студентами шляхом постановки питань та варіантів відповідей. Тести можуть бути створені у

форматі з вибором однієї або кількох правильних відповідей, заповнення пропусків, відповідей відкритого типу та інших. Однією з переваг цього методу є можливість автоматичної перевірки, що забезпечує швидкість та об'єктивність процесу оцінювання.

Ще одним ефективним методом є розв'язання завдань або проектів. Викладачі можуть ставити перед студентами реальні або модельні завдання, які вимагають застосування знань та навичок, отриманих під час навчання. Це може бути написання есе, розв'язання математичних задач, створення проекту або презентації. Оцінювання в цьому випадку здійснюється на основі критеріїв, визначених заздалегідь. Використання цього методу дозволяє викладачам оцінити не лише правильність виконання завдання, але й творчість, креативність та аналітичні здібності студента.

Подальшим методом оцінювання є сприяння дискусіям та взаємодії між студентами у віртуальному навчальному середовищі. Завдяки форумам, чатам та іншим інструментам спілкування, студенти мають можливість обговорювати теми, виражати свої думки та аргументувати свої погляди. Викладач може оцінювати активність та якість участі студентів у таких дискусіях, а також їх здатність до критичного мислення та аргументації.

Окрім розглянутих вище методів, існує ще метод самооцінювання студентів. Самооцінювання є важливим елементом активного навчання та розвитку саморегуляції студентів. У віртуальному навчальному середовищі студенти можуть оцінювати свої власні знання, навички та виробляти плани подальшого розвитку. Це сприяє самосвідомому навчанню та саморефлексії, що позитивно впливає на навчальні результати студентів [18].

1.6. Проблеми платформ онлайн навчання та порівняння їх між собою

Онлайн платформи для навчання мають проблеми, пов'язані з обмеженою взаємодією, відсутністю інтерактивності, недостатнім індивідуальним підходом та недосконалою технічною підтримкою, що впливає на ефективність та задоволеність користувачів.

Недостатня взаємодія зі студентами. Однією з основних проблем платформ для онлайн навчання є відсутність прямого спілкування між викладачами та студентами. Відсутність особистого контакту та можливості задавати питання в режимі реального часу може створювати проблеми у сприйнятті матеріалу та його розумінні.

Багато веб-сайтів та платформ для онлайн навчання пропонують статичний контент без можливості взаємодії або практичних завдань. Це може призвести до недостатнього засвоєння матеріалу студентами та втрати інтересу до навчання.

Оскільки багато платформ пропонують стандартизовані курси, вони не завжди враховують індивідуальні потреби та рівень знань студентів. Це може призвести до того, що деякі студенти відчуються відсталими або незацікавленими через високий темп навчання або недостатній рівень складності матеріалу.

Проблеми з технічною підтримкою. У разі, якщо виникають проблеми з технічними аспектами веб-сайту або платформи, студенти можуть зіткнутися зі складнощами у доступі до навчального контенту або спілкуванні з викладачами. Недостатня технічна підтримка може суттєво ускладнити навчання та порушити навчальний процес.

Відсутність мотивації. Багато платформ навчання не надають достатньої мотивації для студентів. Відсутність конструктивного фідбеку, можливості відстеження прогресу або винагороди за досягнення успіхів може призвести до втрати мотивації студентів та зниження їхньої активності у навчальному процесі.

1.7. Огляд аналогів платформ для онлайн навчання

Існує безліч платформ, які пропонують різноманітні курси та програми для саморозвитку. Було розглянуто декілька популярних платформ онлайн навчання: Udey, Coursera, Skillshare, edX та LinkedIn Learning. Проаналізуємо їх функціонал, визначимо переваги та недоліки кожного з вебсайтів.

1. Udey:

Udey є однією з найпопулярніших платформ онлайн навчання [19]. Вона пропонує величезний вибір курсів у різних галузях знань. Основним перевагою Udey є різноманітність тематик та доступність для всіх користувачів. Багато курсів створюються самими фахівцями у своїй галузі, що дозволяє отримати практичні знання та навички. Кожен курс має власний рейтинг та відгуки студентів, що допомагає вибрати найкращий курс. Однак, Udey не завжди забезпечує високу якість контенту, оскільки всі курси створюються різними по кваліфікації користувачами, і якість може значно варіюватися.

2. Coursera:

Coursera відома своїми курсами, розробленими університетами та високопрофесійними викладачами [20]. На платформі Coursera можна знайти академічні курси, які дозволяють отримати сертифікат від провідних університетів світу. Велика перевага Coursera полягає в тому, що вона пропонує можливість отримати повноцінну освіту онлайн. Курси на Coursera часто мають чітку структуру з лекціями, завданнями та оцінками. Однак, треба відзначити, що більшість курсів на Coursera є платними, хоча можна використовувати їх у безкоштовному режимі без отримання сертифіката.

3. Skillshare:

Skillshare відрізняється від інших платформ тим, що базується на ідеї розвитку навичок творчості та мистецтва [21]. На Skillshare можна знайти відеокурси з графічного дизайну, фотографії, малювання та багато іншого. Основна перевага Skillshare - це практична спрямованість курсів та активна спільнота користувачів,

де можна обговорювати та ділитися власними проектами. Проте, Skillshare має обмежений вибір курсів у порівнянні з іншими платформами, і деякі курси можуть бути менш професійно організованими.

4. edX:

edX є платформою, спільним проектом університетів Массачусетського технологічного і Гарвардського [22]. edX пропонує безкоштовні курси з різних дисциплін, включаючи і академічні. Ця платформа відома високою якістю навчального матеріалу та можливістю отримання сертифікатів від університетів. Курси edX часто мають велику кількість завдань та оцінок, що дозволяє глибше засвоювати матеріал. Однак, деякі користувачі відзначають, що edX може бути складніше для новачків, оскільки матеріал може бути більш академічним і технічним.

5. LinkedIn Learning:

LinkedIn Learning - це платформа, яка поєднує навчання з професійною мережею LinkedIn [23]. На цій платформі можна знайти курси з розвитку різних базових навичок, керівництва, маркетингу та багато іншого. Основна перевага LinkedIn Learning - це можливість вивчати навички, які прямо пов'язані з кар'єрним розвитком та отриманням нових робочих місць. Курси на LinkedIn Learning часто розробляються висококваліфікованими фахівцями та діловими лідерами. Проте, важливо зазначити, що LinkedIn Learning вимагає плати за доступ до своїх курсів.

Кожна з розглянутих платформ має свої переваги та обмеження. Udemy відрізняється широким спектром курсів, Coursera надає академічні програми, Skillshare спрямована на розвиток креативності, edX пропонує якісну академічну освіту, а LinkedIn Learning забезпечує навчання для професійного зростання. Користувачам слід зважати на свої потреби та вподобання, а також розглянути вартість та якість курсів перед вибором платформи для онлайн навчання.

Враховуючи недоліки розглянутих платформ онлайн навчання, існує кілька причин, чому розробка власної спеціалізованої платформи може бути доцільною.

При розробці власної платформи можна забезпечити та спроектувати кращу якість контенту та опрацювати зручний доступ до навчальних матеріалів. На

існуючих платформах, де курси створюються користувачами з різною кваліфікацією, якість матеріалу може бути недостатньою або варіюватися.

Також, розробка власної платформи надає можливість створити унікальну пропозицію, яка відповідає конкретним потребам та вимогам цільової аудиторії. За допомогою власної платформи можна спрямувати увагу на конкретну галузь знань або тип навчання, що дозволить залучити користувачів, які шукають саме такий тип навчання. Можна розробити функції, які дозволяють користувачам вибирати свій власний темп навчання, рівень складності завдань, а також отримувати індивідуальні рекомендації та підтримку.

У разі співробітництва з існуючими платформами, можуть виникати витрати на розміщення курсів, комісійні від продажу та інші витрати. Розробка власної платформи дозволяє контролювати вартість навчання та розподіляти прибуток без залучення третіх сторін.

При створенні такого проекту є можливість організувати сильний бренд та зайняти унікальну позицію на ринку онлайн навчання. Можна розробити власний стиль, дизайн та ідентичність.

Висновки до розділу

У даному розділі були розглянуті ключові аспекти створення платформи для онлайн навчання. Було досліджено бази даних, які є необхідним елементом платформи, що дозволяють зберігати, організовувати та забезпечувати доступ до навчального контенту.

Також, в рамках дослідження були розглянуті питання проектування систем управління для платформи онлайн навчання, що дозволяють ефективно організовувати навчальний процес, адмініструвати користувачів та навчальні ресурси.

Особлива увага була приділена питанням безпеки та конфіденційності в платформах онлайн навчання. Було розглянуто основні загрози та викладені методи захисту від несанкціонованого доступу до навчального контенту та персональних даних користувачів.

Крім того, був розглянутий аспект створення інтерфейсу користувача для платформи онлайн навчання, який забезпечує зручність використання платформи різними категоріями користувачів та пристроями.

Було розглянуто методи оцінювання та контролю знань у віртуальному навчальному середовищі, які дозволяють об'єктивно оцінювати успішність навчання та контролювати рівень засвоєння матеріалу

При дослідженні актуальності завдання були також виявлені проблеми, з якими стикаються платформи онлайн навчання, і проведено порівняння різних платформ з метою виділення їх переваг та недоліків.

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

Для розробки онлайн-платформи для навчання перед початком необхідно провести аналіз, визначившись з інструментами та вимогами до програмного забезпечення. Наступні етапи будуть включені в розробку програмного продукту.

Потрібно визначити набір технологій та інструментів, які найкраще відповідатимуть поставленим вимогам. Це включає в себе мови програмування, фреймворки, бази даних та інші технології, які забезпечать ефективну та надійну роботу платформи.

Далі, проаналізувавши функціональні вимоги до систем управління платформою, які наведено в п. 1.2, будуть використані їх основні концепції по ходу розробки.

На етапі проектування архітектури розробляється загальна архітектура програмного забезпечення. Вона включає розподіл функцій, структуру бази даних, взаємодію з користувачами та інші ключові аспекти системи. Це дозволить зрозуміти, як різні компоненти платформи будуть взаємодіяти між собою.

Далі потрібно визначити алгоритм дій для реалізації програмного продукту. Саме для задачі проектування веб-інтерфейсу були використані наступні етапи такі як:

1. Розробка серверного функціоналу. Написання коду на мові програмування JavaScript у середовищі Node.js для реалізації функціоналу серверної частини платформи.
2. Використання нереляційної бази даних MongoDB: Встановлення та налаштування бази даних MongoDB для зберігання інформації про користувачів, курси та матеріали навчання.
3. Використання простої мови шаблонів handlebars: Написання HTML-шаблонів з використанням простої мови шаблонів handlebars для відображення сторінок платформи.

4. Використання CSS для стилізації сторінок: Застосування CSS для стилізації сторінок платформи та створення привабливого зовнішнього вигляду.

2.1. Вибір інструментів для розробки

Бакалаврська робота була створена за допомогою наступних програм:

- Середовище Visual Studio Code
- Середовище Node.js

Також, згідно з задачами проектування веб-інтерфейсу, були використані відповідні засоби.

Середовище Visual Studio Code

Вибір середовища Visual Studio Code для розробки онлайн платформи для навчання був зумовлений його зручністю та функціональністю, які сприяють ефективній та продуктивній розробці.

Visual Studio Code - це інтегроване середовище розробки, створене компанією Microsoft [24]. Воно надає розширений функціонал для розробки програмного забезпечення і підтримує різноманітні мови програмування. Однією з головних переваг Visual Studio Code є його легкість використання і налаштування. Інтерфейс дуже інтуїтивно зрозумілий. Він має простий дизайн зі зручною розміткою елементів і можливістю налаштовувати його за своїми потребами [25].

На рис. 2.1 показано інтерфейс Visual Studio Code.

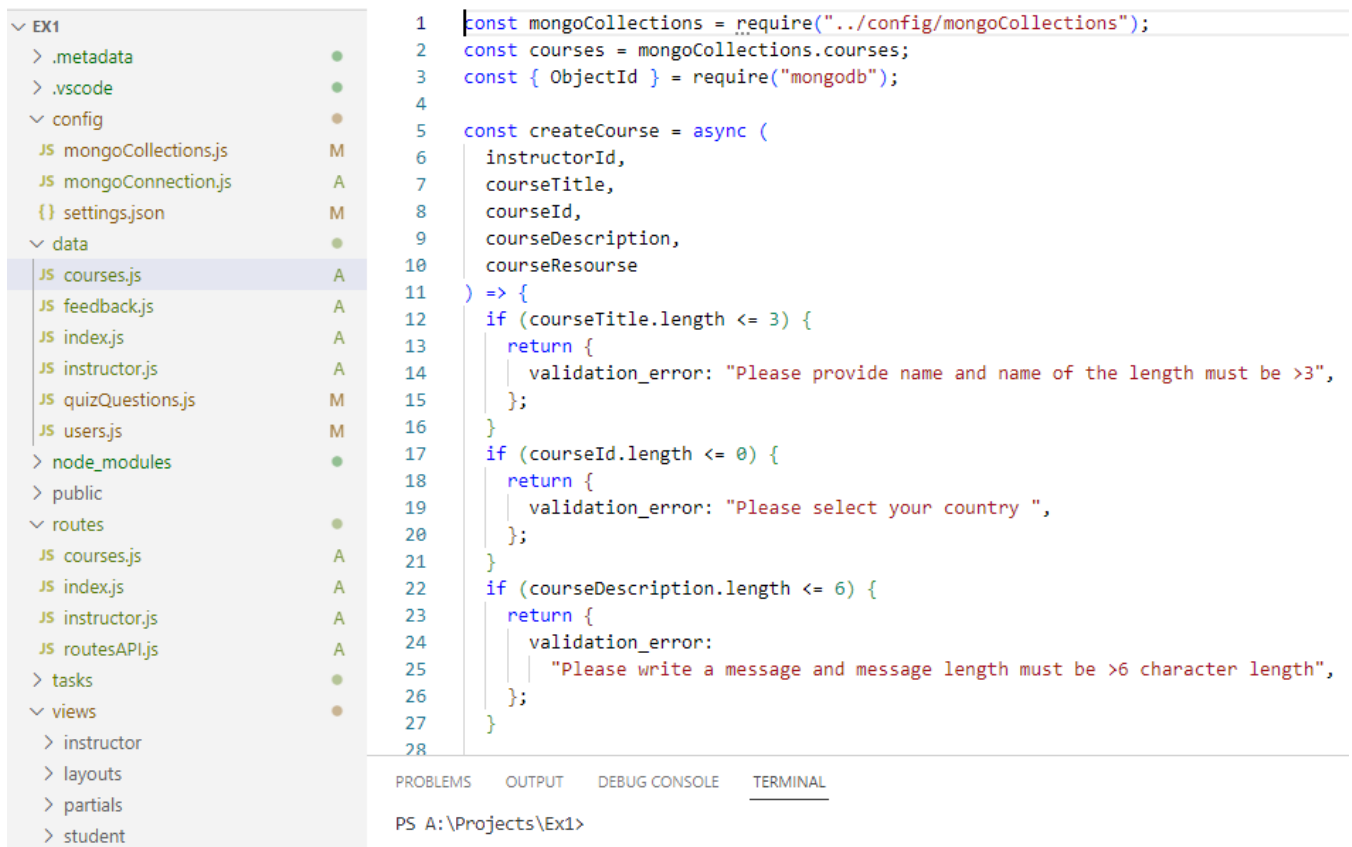


Рис. 2.1. Середовище Visual Studio Code.

Для створення саме веб-інтерфейсу вирішено було використовувати середовище Node.js. Node.js - це потужне та популярне середовище виконання, яке дозволяє використовувати JavaScript на стороні сервера [26]. Node.js використовує механізм V8, розроблений Google, який компілює JavaScript у рідний машинний код і працює з блискавичною швидкістю. Звичайні завдання у веб-додатках, як-от читання або запис у базу даних, читання та запис у мережеві підключення та читання чи запис у файлову систему, можна швидко виконувати за допомогою Node.js. Node.js допомагає підприємствам створювати швидкі, масштабовані мережеві програми, які можуть обробляти одночасні з'єднання з високою пропускнуою здатністю. Node.js нічого не сповільнює; розробникам просто потрібно написати код, і програми Node.js зроблять правильний крок. Наступним інструментом для створення необхідної програми є JavaScript.

JavaScript є високорівневою мовою програмування, яка підтримує різні підходи, такі як імперативний, функціональний і подієво-орієнтований [27]. Її

особливістю є динамічна типізація, а також вона використовується для написання послідовних операцій, які часто називаються "сценаріями" або "скриптами". Такі послідовності зазвичай інтерпретуються, а не компілюються, тому не потребують використання додаткових програм або інструментів для перетворення в інший рівень кодування.

Кожен веб-застосунок або сайт побудований з використанням трьох основних технологій: HTML, CSS і JavaScript. Остання з цих технологій відіграє роль "мозку" розробки і відповідає за інтерактивність та взаємодію з користувачем.

За оформлення контенту відповідають такі інструменти як HTML та CSS (Каскадні таблиці стилів). CSS, також відомий як мова візуального оформлення, є інструментом для представлення вмісту HTML-документа [28, 29]. Його основні функції включають надання кольорів, налаштування відступів, шрифтів та розміщення різних блоків на сторінці. HTML відповідає за організацію інформації та структурування документа, тоді як CSS відповідає за його оформлення шляхом визначення унікальних стилів для кожного елемента. CSS є файлом, в якому описуються правила оформлення всіх елементів на сторінці. Використання CSS дозволяє уникнути повторень коду і полегшує орієнтації в ньому. Один файл стилів можна використовувати для безлічі сторінок, що дозволяє суттєво зменшити час, витрачений на верстку. Кожне правило в файлі CSS складається з селектора і блоку стилів. Селектор визначає, до яких саме елементів документа застосовується правило, а блок стилів містить пари "властивість: значення". Вони розміщуються у фігурних дужках і завершуються крапкою з комою. Коли CSS-файл присутній на сторінці, браузер кешує його, що допомагає зменшити час завантаження сайту при наступних відвідуваннях.

У даній роботі було обрано інший інструмент замість HTML для використання та розробки певних функцій. Цей інструмент називається Handlebars [30]. Вибір був зумовлений певними причинами та міркуваннями, які ускладнюють використання HTML.

Існують дві основні підстави для створення шаблону для веб-сайту. По-перше, створення шаблону спонукає до відокремлення логічного коду від фактичного

представлення, допомагаючи дотримуватися шаблону Вигляд/Контролер. По-друге, шаблони зберігають код чистим і підтримуваним, що в свою чергу робить процес оновлення веб-сайту швидким. Користувач не створює веб-сайт з використанням Handlebars. Замість цього створюються рекомендації і структури, які визначають зовнішній вигляд веб-сайту, не зосереджуючись на конкретних даних сторінки.

Розглянемо деякі основні аспекти. Handlebars генерує код HTML, використовуючи структуру JSON і запускаючи її через шаблон. Ці шаблони в основному написані на звичайному HTML і заповнені функціями, які дозволяють вводити дані за потреби. Наприклад, наступний шаблон призначений для вставки даних про курс (заголовок, ідентифікатор та опис) в HTML-код (рис. 2.2).

```
<div class="card-header">{{enrolled.courseTitle}} ({{this.courseId}})</div>  
<div class="card-body">{{enrolled.courseDescription}}</div>
```

Рис. 2.2. Приклад коду Handlebars

У першому `<div>` використовуються дві змінні, обгорнуті в двох фігурних дужках: `{{enrolled.courseTitle}}` та `{{this.courseId}}`. Значення цих змінних будуть вставлені на місце цих виразів під час генерації HTML-коду. `enrolled.courseTitle` містить заголовок курсу, а `this.courseId` містить ідентифікатор курсу.

У другому `<div>` використовується змінна `{{enrolled.courseDescription}}`, яка містить опис курсу. Її значення також буде вставлене в HTML-код під час генерації.

2.2. Функціональність та особливості MongoDB онлайн-платформи.

Для збору інформації про користувачів та їх ID було використано базу даних MongoDB. Аналіз даних відбувається за допомогою GUI MongoDB Compass.

MongoDB Compass - це офіційний графічний інтерфейс для роботи з MongoDB, який дозволяє зручно взаємодіяти з базами даних [31]. MongoDB Compass має сучасний та інтуїтивно зрозумілий інтерфейс, який наведено на рис. 2.3, що дозволяє легко керувати та виконувати операції над базами даних MongoDB.

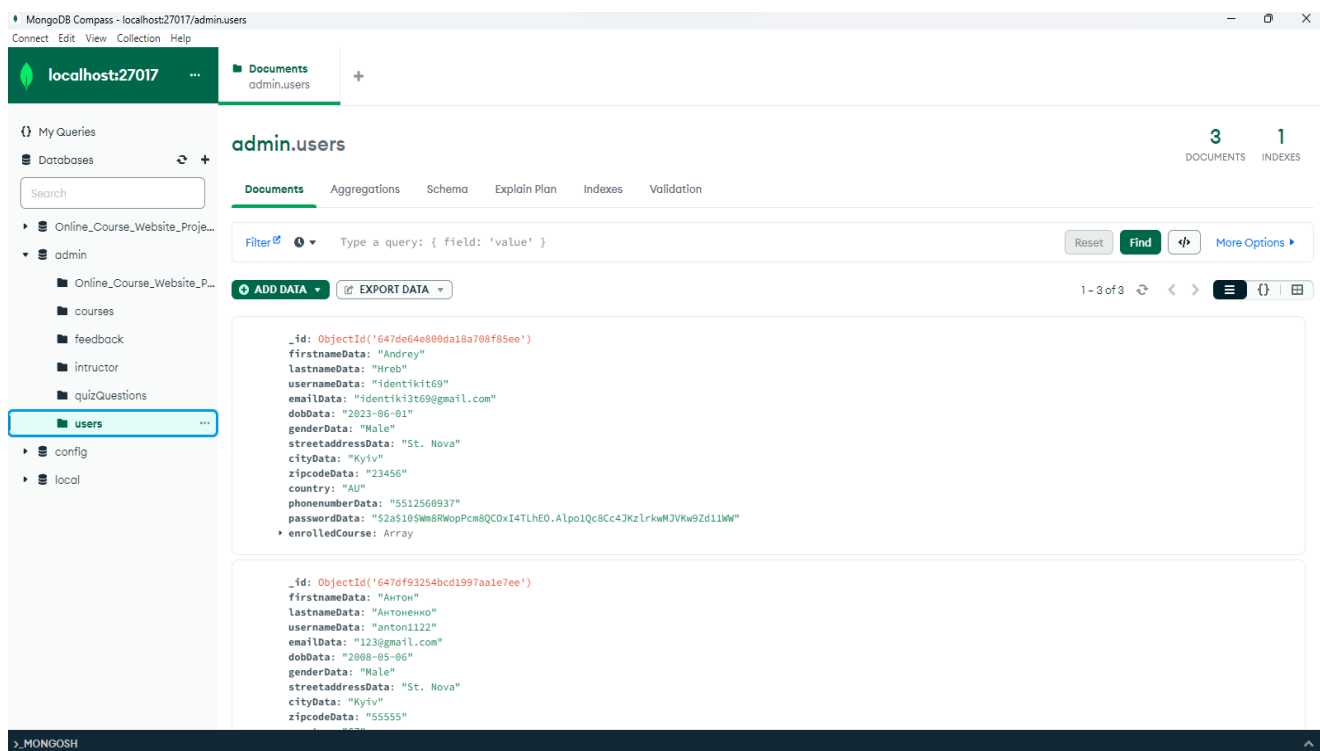


Рис. 2.3. Графічний інтерфейс MongoDB Compass

При здійсненні підключення та взаємодії з базою даних MongoDB можна виділити наступні етапи:

- Підключення до сервера.
- Отримання об'єкта бази даних на сервері.
- Отримання об'єкта колекції в базі даних.

- Взаємодія з колекцією (додавання, видалення, отримання, зміна даних).

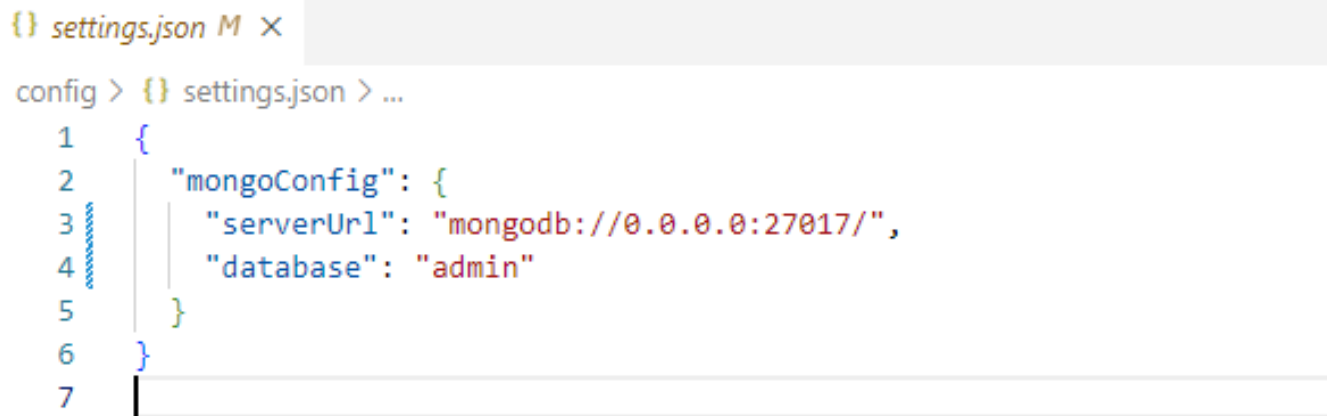
Код, що наведено на рис. 2.4, представляє модуль, який надає функціональність для підключення та відключення від бази даних MongoDB.

```
JS mongoConnection.js A X
config > JS mongoConnection.js > ...
1  const MongoClient = require('mongodb').MongoClient;
2  const settings = require('./settings');
3  const mongoConfig = settings.mongoConfig;
4
5  let _connection = undefined;
6  let _db = undefined;
7
8  module.exports = {
9    dbConnection: async () => {
10     if (!_connection) {
11       _connection = await MongoClient.connect(mongoConfig.serverUrl);
12       _db = await _connection.db(mongoConfig.database);
13     }
14
15     return _db;
16   },
17   closeConnection: () => {
18     _connection.close();
19   },
20 };
21
```

Рис. 2.4. Власний програмний код підключення та відключення баз даних MongoDB

Значення MongoClient використовується для створення клієнта MongoDB, а settings, код якого наведено на рис. 2.5, містить конфігураційні дані, які використовуються для налаштування підключення до сервера MongoDB.

Змінні _connection та _db ініціалізуються як undefined і використовуються для зберігання постійного з'єднання з базою даних та об'єкта бази даних відповідно.



```

{} settings.json M X
config > {} settings.json > ...
1  {
2    "mongoConfig": {
3      "serverUrl": "mongodb://0.0.0.0:27017/",
4      "database": "admin"
5    }
6  }
7

```

Рис. 2.5. Власний програмний код підключення до сервера MongoDB

Функція `dbConnection` є асинхронною і перевіряє, чи вже встановлене з'єднання з базою даних. Якщо з'єднання відсутнє, вона виконує підключення до сервера MongoDB за допомогою `MongoClient.connect()`, використовуючи URL сервера з конфігурації. Потім вона отримує об'єкт бази даних, використовуючи `_connection.db()`, зазначаючи назву бази даних з конфігурації. Результат зберігається у `_db` і повертається як результат функції.

Функція `closeConnection` відповідає за закриття з'єднання з базою даних за допомогою `_connection.close()`.

Щоб дані, які вводить користувач були занесені в базу даних, при цьому, маючи різні каталоги, було представлено таке рішення, яке наведено на рис. 2.6.

Даний код визначає модуль, який надає функції для отримання колекцій з бази даних MongoDB.

Функція `getCollectionFn` приймає параметр `collection` (назва колекції) і повертає асинхронну функцію. Внутрішня змінна `_col` ініціалізується як `undefined`.

Повернена асинхронна функція перевіряє, чи вже отримана колекція з бази даних. Якщо `_col` є `undefined`, вона отримує доступ до об'єкта бази даних, викликаючи функцію `dbConnection` з модуля `dbConnection`. Потім вона отримує об'єкт колекції, використовуючи `_col = await db.collection(collection)`. Результат зберігається у `_col` і повертається як результат функції.

```

config > JS mongoCollections.js > ...
1  const dbConnection = require("../mongoConnection");
2  const getCollectionFn = (collection) => {
3    let _col = undefined;
4
5    return async () => {
6      if (!_col) {
7        const db = await dbConnection.dbConnection();
8        _col = await db.collection(collection);
9      }
10
11     return _col;
12   };
13 };
14 module.exports = {
15   users: getCollectionFn("users"),
16   instructor: getCollectionFn("instructor"),
17   courses: getCollectionFn("courses"),
18   feedback: getCollectionFn("feedback"),
19   quizQuestions: getCollectionFn("quizQuestions"),
20 };
21

```

Рис. 2.6. Власний програмний код для отримання колекцій з баз даних

Останній рядок модуля експортує об'єкт, який містить властивості з функціями отримання колекцій. Кожна властивість має назву колекції і використовує функцію `getCollectionFn` для отримання відповідної колекції з бази даних.

Тобто, використавши ці модулі, конкретна інформація на сайті, яку вводить користувач, буде автоматично занесена в базу даних. Прикладом цього є дані користувача, так як: Прізвище та ім'я, нікнейм користувача, адрес електронної пошти, дата народження, стать, адреса проживання, місто, поштовий індекс, країна або регіон, номер телефону та пароль. Як це виглядає, показано на рис. 2.7.

```

_id: ObjectId('647de8d9800da18a708f85ef')
firstnameData: "Andrey"
lastnameData: "Admin"
usernameData: "admin"
emailData: "idntikit69@gmail.com"
dobData: "2023-06-01"
genderData: "Male"
streetaddressData: "St. Nova"
cityData: "Kyiv"
zipcodeData: "23456"
country: "CN"
phonenumberData: "1234567890"

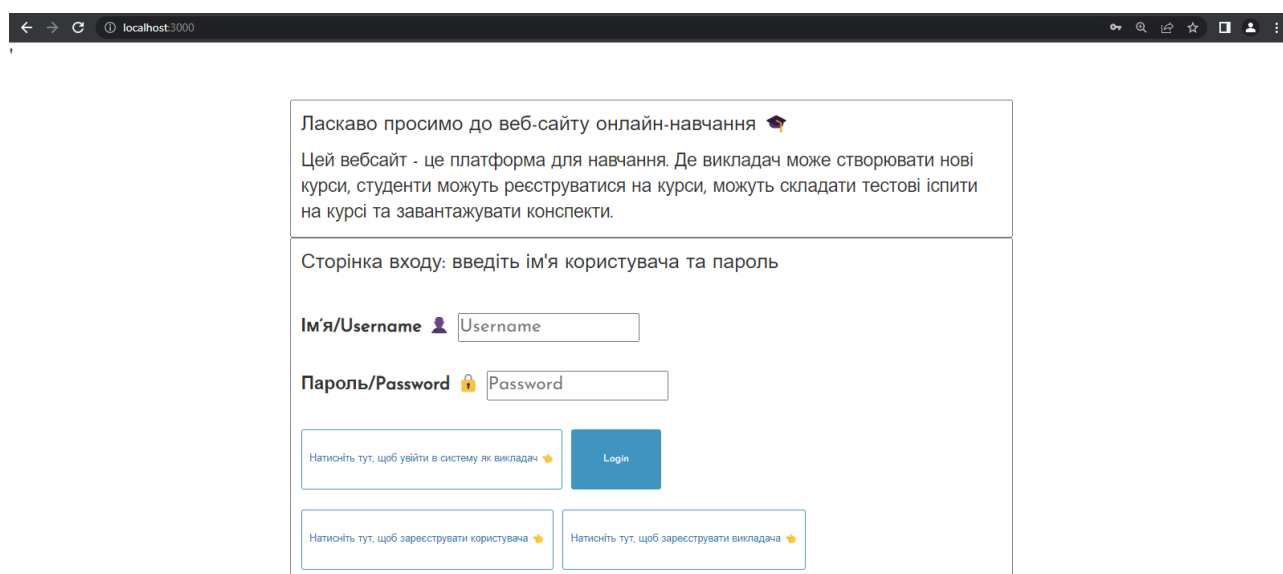
```

Рис. 2.7. Дані користувача в базі даних

2.3. Опис інтерфейсу та його функціонал

Веб-інтерфейс для онлайн навчання, розглянутий у цьому підрозділі, має на меті забезпечити зручну та ефективну платформу для навчання, де користувачі можуть отримувати якісні знання та вдосконалювати свої навички. Використання сучасних технологій, правильне проектування інтерфейсу та функціональних можливостей допоможуть забезпечити успішне та зручне навчання в онлайн середовищі.

Початкова сторінка веб-додатку має функцію реєстрації користувача. Інтерфейс вводу даних (реєстрація та вхід як зареєстрований користувач) виглядає наступним чином (рис. 2.8).



The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page content is as follows:

Ласкаво просимо до веб-сайту онлайн-навчання 🎓

Цей вебсайт - це платформа для навчання. Де викладач може створювати нові курси, студенти можуть реєструватися на курси, можуть складати тестові іспити на курсі та завантажувати конспекти.

Сторінка входу: введіть ім'я користувача та пароль

Ім'я/Username 

Пароль/Password 

Рис. 2.8. Початкова сторінка веб-додатку

При реєстрації, користувача направляє до сторінки, де потрібно ввести всю потрібну інформацію (рис. 2.9).

Сторінка реєстрації користувача

Ім'я/First Name* : Прізвище/Last Name* :

Нікнейм користувача/Enter Username* :

Введіть електронну пошту/Enter your email* :

Дата народження/Birthday: 

Оберіть стать/Choose your gender:

Адреса проживання/Street address :

Місто/City :

Поштовий індекс/ZIP or postal code(optional) :

Країна або регіон/Country or region :

Введіть номер телефону/Enter a phone number:

Створіть пароль/Create Password* :

[Натисніть тут, щоб переглянути сторінку реєстрації викладача](#) ➡

Рис. 2.9. Сторінка реєстрації користувача

Вся введена інформація буде автоматично зібрана і відсортована у локальній базі даних у окремому каталозі. Кожен каталог відповідає за свій набір даних.

Програмний код для правильної роботи введення, перевірки на правильність вводу інформації буде вказано в Додатку А.

Вигляд головної сторінки веб-сайту наведено на рис. 2.10

Веб-сайт онлайн-курсів Головна Тестування

- Головна
- Профіль
- Курси
- Відгук/Допомога
- Зв'язок
- Про Веб-сайт
- Вийти

identikit69's Enrolled Courses.

Java (java)

<https://www.javatpoint.com/java-basics>

Рис 2.10. Головна сторінка веб-додатку

Вгорі сторінки розташована назва сайту, яка слугує посиланням на головну сторінку.

Поряд з назвою розташовано посилання на профіль користувача. Натиснувши на це посилання, користувач буде направлений на сторінку свого особистого профілю, де потім можна буде його редагувати. Ці сторінки наведені на рис. 2.11-2.12.

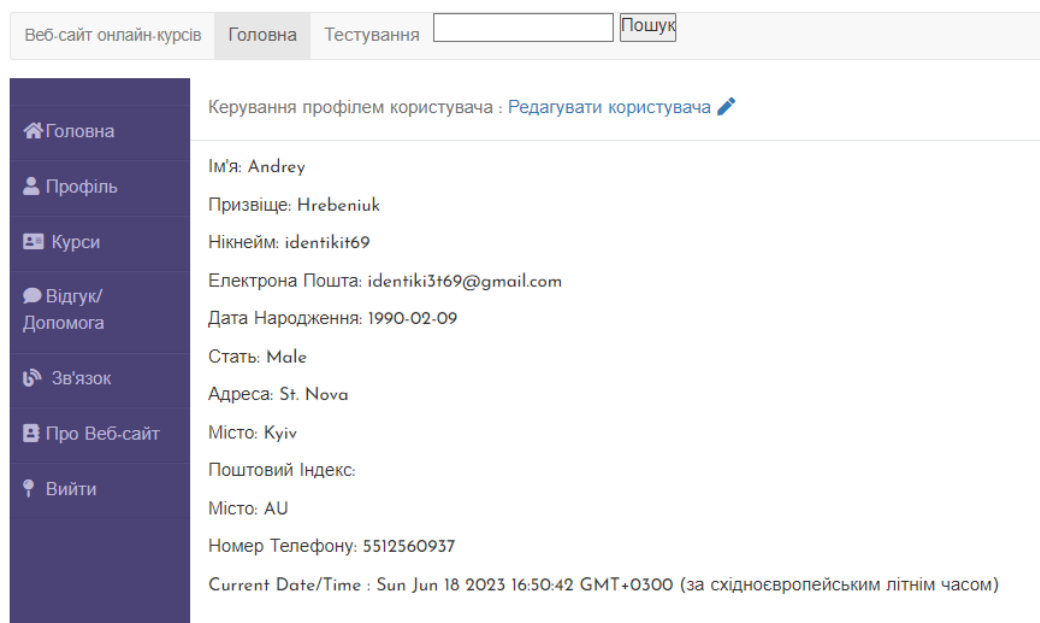


Рис. 2.11. Сторінка профілю користувача.

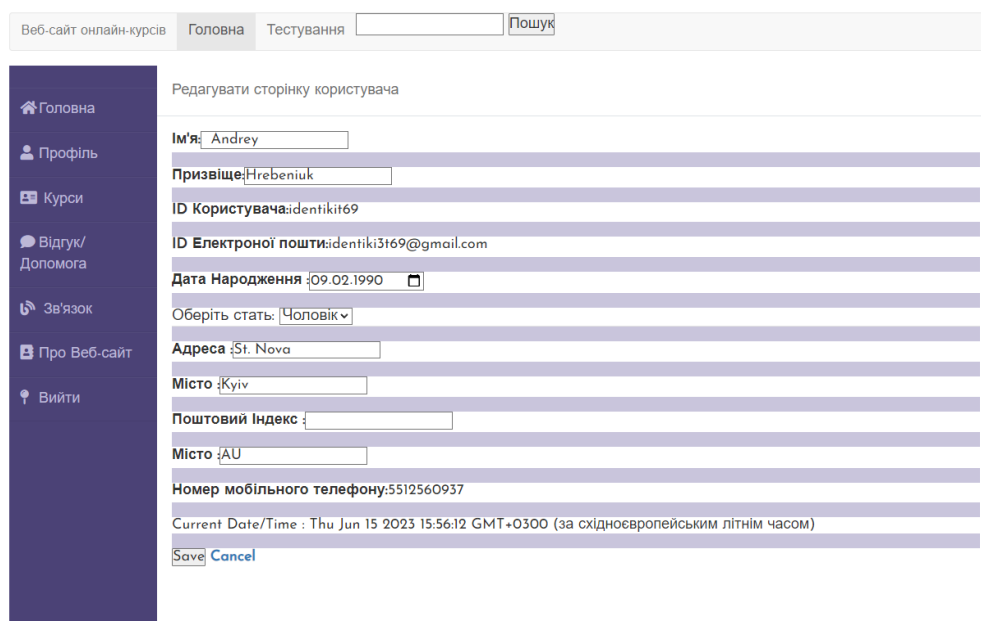


Рис. 2.12. Сторінка редагування інформації про користувача.

Також на головній сторінці розміщено посилання на список курсів. Натиснувши на це посилання, користувач може переглянути доступні курси (рис. 2.13).

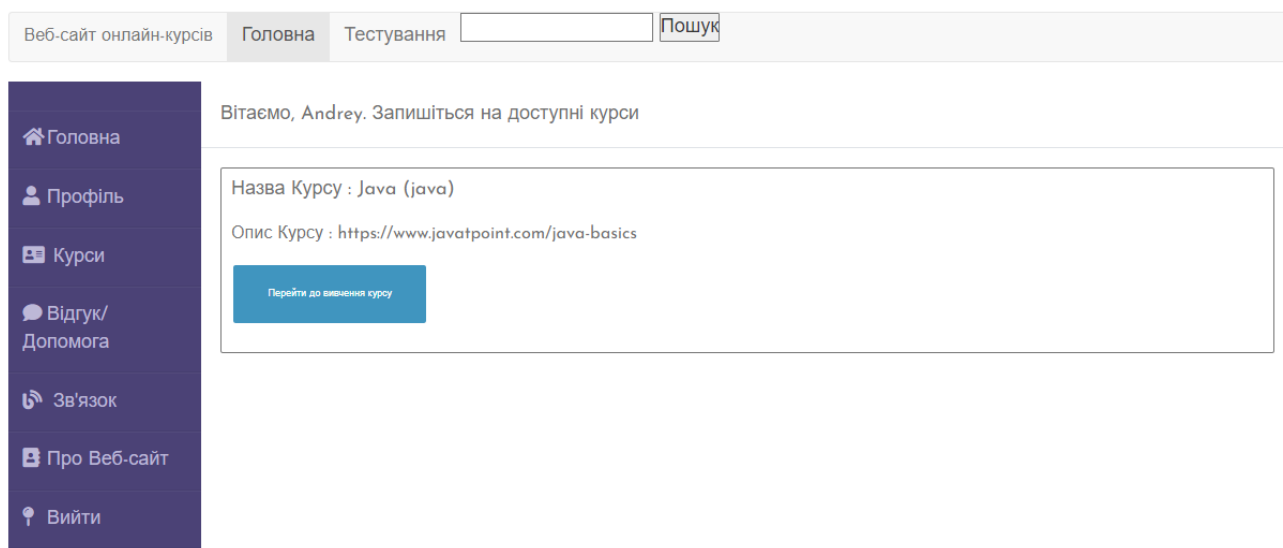


Рис. 2.13. Сторінка вибору курсів

Є також посилання на сторінку, яку показано на рис. 2.14, де користувач може залишити відгук про курс, який він проходить або вже завершив. Це посилання допоможе користувачам спілкуватися та ділитися своїми враженнями.

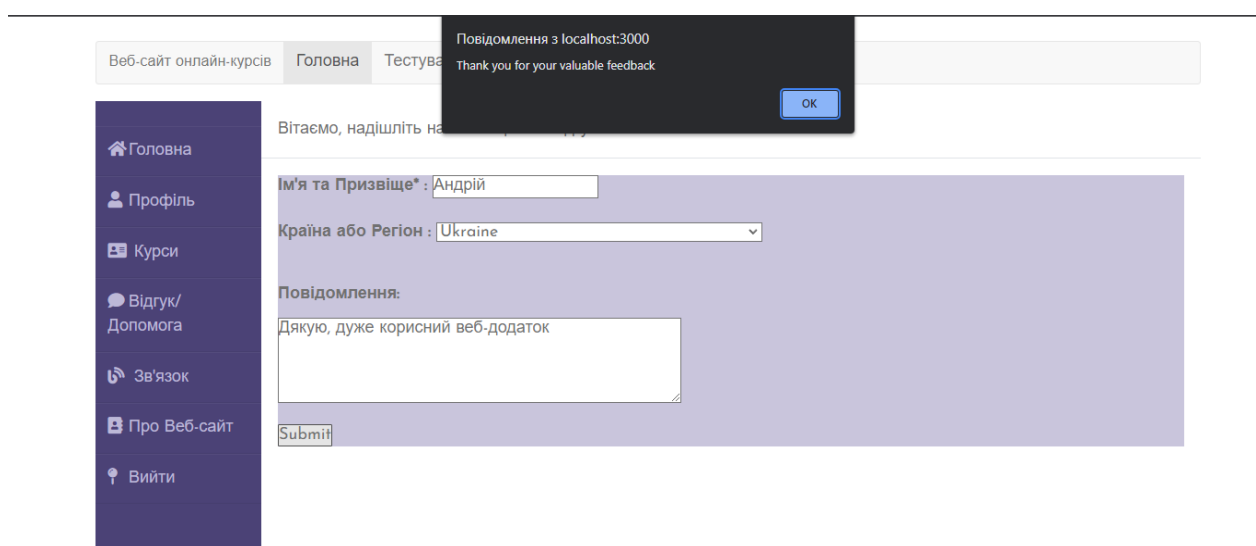


Рис. 2.14. Заповнення форми для відгуку.

Цей відгук буде автоматично занесений до бази даних (рис. 2.15).

admin.feedback

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' }

ADD DATA **EXPORT DATA**

```

_id: ObjectId('648b0ba49c74ce0512256d7f')
feedBackId: "647de64e800da18a708f85ee"
fullname: "Андрій"
country: "UA"
message: "Дякую, дуже корисний веб-додаток"

```

Рис. 2.15. Інформація, яка з'явилася у базі даних

Для забезпечення зручності користувачів, на головній сторінці також розташована контактна інформація. Ця сторінка, яку наведено на рис. 2.16, містить дані для зв'язку з адміністрацією сайту, наприклад, соціальні мережі.

Веб-сайт онлайн-курсів Головна Тестування

- Головна
- Профіль
- Курси
- Відгук/
Допомога
- Зв'язок
- Про Веб-сайт
- Вийти

Облікові записи профілів у соціальних мережах

Слідкувати за нами на Facebook

Слідкувати за нами в Instagram

Слідкувати за нами на Youtube

Рис. 2.16. Сторінка для зв'язку з адміністрацією

Крім того, на головній сторінці розміщена коротка інформація про веб-сайт, яка пояснює його ціль та мету. Це допомагає відвідувачам зрозуміти, які можливості надає ця платформа для онлайн навчання.

Нарешті, знизу сторінки є посилання на вихід з акаунту. Користувач може натиснути на нього, щоб вийти зі свого облікового запису та повернутися до головної сторінки.

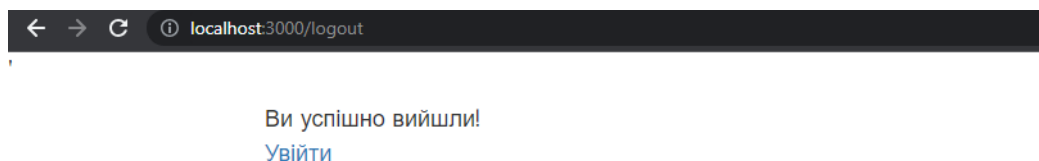


Рис. 2.17. Вихід з акаунту користувача

Сторінку викладача для створення курсу наведено на рис. 2.18. На цій сторінці розміщене поле введення для назви курсу. Викладач може ввести тут назву, що якнайкраще описує зміст курсу.

Також на сторінці викладача є поле введення для ID курсу. Це унікальний ідентифікатор, який допомагає відрізнити курси один від одного. Викладач може ввести тут бажаний ID.

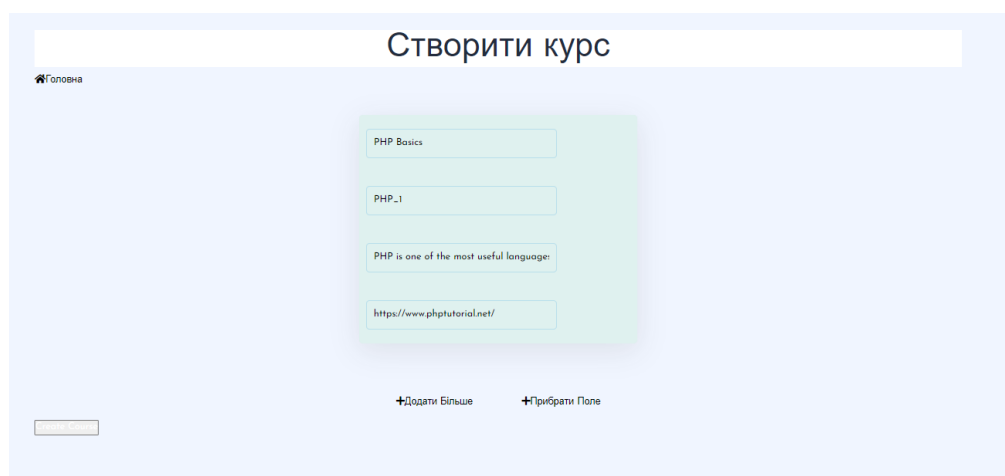


Рис. 2.18. Сторінка створення курсу

Крім того, на сторінці присутнє поле введення для опису курсу. Викладач може коротко описати зміст, мету та особливості курсу.

Після заповнення усіх необхідних полів, викладач має можливість зберегти створений курс та розпочати його редагування, додаванням матеріалів, завдань та іншого змісту. Поряд з цією функцією, викладач може створити тест, спираючись на курс.

Веб-сторінка для створення тестів на платформі онлайн навчання надає можливість викладачам зручно створювати тести для своїх курсів. На цій сторінці викладач може вводити необхідну інформацію, пов'язану з тестом, який він бажає створити (рис. 2.19)

Інтерфейс сторінки містить наступні поля:

ID курсу: Викладач вводить ідентифікатор курсу, до якого будуть прив'язані запитання тесту. Це допомагає упорядкувати тести і легко знаходити їх, коли потрібно.

Запитання: Викладач вводить текст запитання для тесту. Тут він може пояснити суть запитання або вказати будь-яку необхідну контекстну інформацію.

Варіанти відповідей: Викладач вводить чотири варіанти відповідей для кожного запитання. Кожен варіант відповіді повинен бути коротким текстом, що точно відображає можливу відповідь на запитання.

The image shows a web form titled "Додати Питання" (Add Question). It contains several input fields for creating a test question. The fields are labeled as follows: "Введіть ID Курсу" (Enter Course ID), "Введіть Запитання" (Enter Question), "Введіть варіант 'a'", "Введіть варіант 'b'", "Введіть варіант 'c'", "Введіть варіант 'd'", and "Введіть правильну відповідь на запитання" (Enter the correct answer to the question). At the bottom of the form is a blue button labeled "Додати Запитання" (Add Question).

Рис. 2.19. Вигляд сторінки для створення тестів

Правильна відповідь: Викладач визначає правильну відповідь на запитання, обираючи відповідну літеру (a, b, c, d) у відповідному полі. Це дозволяє системі правильно оцінювати відповіді учнів.

Після того, як викладач ввів всю необхідну інформацію, відбувається автоматичне занесення даних про тест до бази даних платформи. Це забезпечує зручний доступ до тесту та його результатів, як для викладача, так і для учнів, які проходять курс (рис. 2.20).

ADD DATA EXPORT DATA

```
_id: ObjectId('647dea91800da18a708f85f2')
courseId: "Java1"
quizQuestions: Array
  0: Object
    question: null
    a: "Programming Language"
    b: "IDE"
    c: "Web"
    d: "None"
    correctAnswer: "a"
```

Рис. 2.20. Інформація про тест у базі даних.

Завдяки зручному інтерфейсу сторінки для створення тесту викладачі можуть ефективно та швидко створювати цікаві тести для своїх студентів, зберігаючи всю необхідну інформацію у системі онлайн навчання.

Висновки до розділу

У даному розділі були досліджені та описані ключові аспекти засобів розробки програмного продукту. В п. 2.1 було проведено вибір необхідних інструментів для розробки, який відповідає потребам проекту та дозволяє ефективно втілити функціонал продукту. Також, була проведена характеристика функціональності та особливостей системи бази даних, що використовується у

програмному продукті. Були визначені вимоги до бази даних, а також викладені особливості її структури та функцій.

У п. 2.3 було описано розроблений інтерфейс програмного продукту та його функціонал. Були розглянуті вимоги до інтерфейсу, його зовнішній вигляд та навігаційна структура. Також було описано функціонал, який надає користувачеві можливість здійснювати необхідні дії та використовувати функціональні можливості продукту.

ВИСНОВКИ

У цій дипломній роботі була поставлена задача створення платформи для онлайн навчання. Було досліджено різні аспекти, пов'язані з розробкою такої платформи, включаючи бази даних, системи управління, питання безпеки та конфіденційності, інтерфейс користувача, методи оцінювання та контролю знань, проблеми та порівняння платформ, а також проведений огляд аналогів.

У першому розділі досліджено різні типи баз даних, зокрема реляційні, NoSQL та графові бази даних, та їх застосування у платформах для онлайн навчання. Було виявлено, що вибір певного типу бази даних залежить від конкретних потреб платформи та її функціональності.

Також розглянуто проектування систем управління для платформ онлайн навчання. Були визначені ключові аспекти, що впливають на розробку таких систем, зокрема функціональні вимоги, масштабованість, надійність та швидкодію.

Також були розглянуті питання безпеки та конфіденційності в платформах онлайн навчання. Виявлено, що ці аспекти є критичними для успішної реалізації платформи, оскільки забезпечують захист особистих даних користувачів та запобігають несанкціонованому доступу до навчального контенту.

Аналіз інтерфейсу користувача для платформи онлайн навчання надав можливість визначити основні принципи проектування зручного та ефективного інтерфейсу, а також вимоги до його функціоналу.

Проведено огляд аналогів платформ для онлайн навчання. Цей огляд надає змогу ознайомитися з існуючими рішеннями та використати їх досвід при розробці власної платформи.

У другому розділі розглянуто розробку програмного продукту, включаючи вибір необхідних інструментів, опис функціональності та особливостей системи бази даних, а також опис розробленого інтерфейсу та його функціоналу.

Отже, на основі проведеного дослідження можна зробити висновок, що була успішно виконана поставлена задача створення платформи для онлайн навчання. В

роботі розглянуті ключові аспекти, що впливають на її розробку, і виконано необхідні кроки для створення функціональної та ефективної платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] The 10 Benefits of Online Learning - School of Education. School of Education : вебсайт. URL: <https://drexel.edu/soe/resources/student-teaching/advice/benefits-of-online-and-virtual-learning/> (дата звернення: 10.06.2023).

[2] 10 Surprising Benefits of Online Learning. Coursera. : вебсайт. URL: <https://www.coursera.org/articles/benefits-of-online-learning> (дата звернення: 10.06.2023).

[3] Clark, R. C., & Mayer, R. E. (2016). E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning

[4] Reigeluth, C. M., & Carr-Chellman, A. A. (Eds.). (2009). Instructional-design theories and models: Building a common knowledge base. Routledge.

[5] Moore, M. G., & Kearsley, G. (2011). Distance education: A systems view of online learning. Cengage Learning.

[6] S. Kuznets KhNUE Personal Learning Systems: вебсайт. URL: https://pns.hneu.edu.ua/pluginfile.php/601514/mod_resource/content/1/Лекція%20%20Реляційні%20бази%20даних.pdf. (дата звернення: 10.06.2023).

[7] SQL база даних Для чого призначена база даних?. ukraine.com.ua: вебсайт. URL: <https://www.ukraine.com.ua/uk/blog/programming/sql-baza-dannih-dlya-chego-prednaznachena-baza-dannih.html> (дата звернення: 11.06.2023).

[8] What Is NoSQL? NoSQL Databases Explained. MongoDB: вебсайт. URL: <https://www.mongodb.com/nosql-explained> (дата звернення: 11.06.2023).

[9] Учасники проектів Вікімедіа. Графова база даних – Вікіпедія. Вікіпедія: вебсайт. URL: https://uk.wikipedia.org/wiki/Графова_база_даних (дата звернення: 11.06.2023).

[10] MongoDB: The Developer Data Platform. MongoDB. URL: <https://www.mongodb.com> (дата звернення: 19.06.2023).

[11] JSON And BSON. MongoDB: вебсайт. URL: <https://www.mongodb.com/json-and-bson> (дата звернення: 11.06.2023).

[12] Git - Про систему контролю версій. Git: вебсайт. URL: <https://git-scm.com/book/uk/v2/Вступ-Про-систему-контролю-версій> (дата звернення: 12.06.2023).

[13] Що таке функціональні вимоги: приклади, визначення, повний посібник: вебсайт. URL: <https://visuresolutions.com/uk/blog/functional-requirements> (дата звернення: 19.06.2023).

[14] Доценко С., Ворожбіт-Горбатюк В., Собченко Т. ОНЛАЙН-БЕЗПЕКА УЧАСНИКІВ ОСВІТНЬОГО ПРОЦЕСУ В УМОВАХ ДИСТАНЦІЙНОГО І ЗМІШАНОГО НАВЧАННЯ: вебсайт. URL: <https://znayshov.com/FR/11716/545.pdf>. (дата звернення: 12.06.2023).

[15] Makris N. Creating Adaptable and Adaptive User Interface Implementations in Model Driven Developed Software // Radboud University Nijmegen Press. 2014. № 3, pp.8-12. (дата звернення: 12.06.2023).

[16] Gullà F., Ceccacci S. Design Adaptable and Adaptive User Interfaces: a Method to Manage the Information // Ambient Assisted Living: Italian Forum 2014. 2014. № 3, pp.2-4. (дата звернення: 12.06.2023).

[17] А. А. Бугай В. В. Олійник. КОЦЕПТУАЛЬНА МОДЕЛЬ АДАПТИВНОГО ВЕБ-ІНТЕРФЕЙСУ КОРИСТУВАЧА З ВИКОРИСТАННЯМ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ: вебсайт. URL: https://ela.kpi.ua/bitstream/123456789/34173/1/asau-2018-1_02.pdf. (дата звернення: 11.06.2023).

[18] ПРОБЛЕМИ ОЦІНЮВАННЯ НАВЧАЛЬНИХ ДОСЯГНЕНЬ УЧНІВ В УМОВАХ ДИСТАНЦІЙНОГО НАВЧАННЯ В ПЕРІОД ПАНДЕМІЇ. Welcome to - Digital Library NAES of Ukraine: вебсайт. URL: <https://lib.iitta.gov.ua/721501/1/tezy2.pdf>. (дата звернення: 15.06.2023).

[19] Udemy: Online Courses - Learn Anything, On Your Schedule: вебсайт. URL: <https://www.udemy.com> (дата звернення: 13.06.2023).

[20] Coursera | Degrees, Certificates, & Free Online Courses: вебсайт. URL: <https://www.coursera.org> (дата звернення: 13.06.2023).

[21] Skillshare: Online Classes for Creatives: вебсайт. URL: <https://www.skillshare.com> (дата звернення: 13.06.2023).

[22] edX | Build new skills. Advance your career. | edX: вебсайт. URL: <https://www.edx.org> (дата звернення: 13.06.2023).

[23] LinkedIn Learning: вебсайт. URL: <https://www.linkedin.com/learning> (дата звернення: 11.06.2023).

[24] Microsoft: вебсайт. URL: <https://www.microsoft.com> (дата звернення: 19.06.2023).

[25] Microsoft. Visual Studio Code - Code Editing. Redefined. Visual Studio Code - Code Editing. Redefined: вебсайт. URL: <https://code.visualstudio.com> (дата звернення: 19.06.2023).

[26] 10 reasons why you should use NodeJs. ProjectPro: вебсайт. URL: <https://www.projectpro.io/article/10-reasons-why-you-should-use-nodejs/129> (дата звернення: 19.06.2023).

[27] Чому JavaScript — перспективна мова програмування? Поради початківцям: вебсайт. URL: <https://dou.ua/forums/topic/35184/> (дата звернення: 19.06.2023).

[28] Що таке CSS? вебсайт. URL: <https://avada-media.ua/ua/css/> (дата звернення: 19.06.2023).

[29] Contributors to Wikimedia projects. HTML - Wikipedia. Wikipedia, the free encyclopedia: вебсайт. URL: <https://en.wikipedia.org/wiki/HTML> (дата звернення: 19.06.2023).

[30] Manricks G. Введение в Handlebars. Code Envato Tuts: вебсайт. URL: <https://code.tutsplus.com/ru/tutorials/--handlebars--net-27761> (дата звернення: 19.06.2023).

[31] MongoDB Compass. MongoDB. URL: <https://www.mongodb.com/products/compass> (дата звернення: 19.06.2023).

Додаток А

Функція введення інформації та її перевірка на правильність

Текст програми

Аркушів 15

Програмний код для функції введення інформації та її перевірка на правильність:

```
const mongoCollections = require("../config/mongoCollections");
const users = mongoCollections.users;
const { ObjectId } = require("mongodb");
var bcrypt = require("bcryptjs");
var salt = bcrypt.genSaltSync(10);
const coursess = require("../data/courses");
```

```
function ValidateEmail(email_id) {
  const regex_pattern =
    /^((([^<>()[\]\\\.,;:\s@\"']+)(\.[^<>()[\]\\\.,;:\s@\"']+)+)*)(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.([a-zA-Z]{2,}))$|
```

```
  return regex_pattern.test(email_id);
}
```

//check phone

```
function ValidatePhonenumber(phone) {
  const regex = /^[+]?([0-9]{3})?[-\s\.]?[0-9]{3}[-\s\.]?[0-9]{4,6}$/im;
  return regex.test(phone);
  //console.log(regex.test(phone))
}
```

//space check

```
function hasWhiteSpace(s) {
  return s.indexOf(" ") >= 0;
}
```

//alpha numeric

```
function onlyLettersAndNumbers(str) {
```

```
    return /^[A-Za-z0-9]*$/ .test(str);
  }
}
```

//checking Password must have at least one uppercase character, one number and one special character.

```
function checkPassword(str) {
  return /^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[^\w\s]).{6,}$/ .test(str);
}
```

```
const createUser = async (
  firstnameData,
  lastnameData,
  usernameData,
  emailData,
  dobData,
  genderData,
  streetaddressData,
  cityData,
  zipcodeData,
  country,
  phonenumberData,
  passwordData
) => {
  //validate data first
  usernameData = usernameData.toLowerCase();
  emailData = emailData.toLowerCase();
  if (firstnameData.length <= 0) {
    return {
      validation_error: "The length of the first must be greater than 0! ",
    };
  }
}
```

```

if (lastnameData.length < 4) {
  return {
    validation_error: "You have entered an invalid phoone number! ",
  };
}
if (usernameData.length < 4) {
  return {
    validation_error: "The length of the Username must be greater than 4 ",
  };
}
if (hasWhiteSpace(usernameData)) {
  return {
    validation_error: "Username must not contain space character",
  };
}
if (onlyLettersAndNumbers(usernameData) == false) {
  return {
    validation_error: "Username must be Alphanumeric",
  };
}
if (checkPassword(passwordData) == false) {
  return {
    validation_error:
      "Password must have at least one uppercase character, one number and one
special character.",
  };
}
if (hasWhiteSpace(passwordData)) {
  return {
    validation_error: "Password must not contain space character",
  };
}

```

```

    };
}
if (passwordData.length < 6) {
    return {
        validation_error: "The length of the Password must be greater than 6 ",
    };
}
if (ValidateEmail(emailData) == false) {
    return {
        validation_error: "You have entered an invalid email address! ",
    };
}

if (ValidatePhonenumber(phonenumberData) == false) {
    return {
        validation_error: "You have entered an invalid phone number! ",
    };
}

//check phone number

//verify user does not exist in db
doesUserExist = await getUserByUsername(usernameData);

doesPhoneExist = await getUserByPhone(phonenumberData);
doesEmailExist = await getUserByEmail(emailData);

if (doesPhoneExist != null) {
    return {
        validation_error:

```

"Please provide another Phone number, the given phone number address already exists! ",

```
};
}
```

```
if (doesEmailExist !== null) {
```

```
  return {
```

```
    validation_error:
```

```
      "Please provide another email address, the given email address already exists! ",
```

```
    };
  }
```

```
if (doesUserExist === null) {
```

```
  //user does not exist, therefore save record
```

```
  let new_user;
```

```
  new_user = {
```

```
    firstNameData: firstNameData,
```

```
    lastNameData: lastNameData,
```

```
    usernameData: usernameData,
```

```
    emailData: emailData,
```

```
    dobData: dobData,
```

```
    genderData: genderData,
```

```
    streetaddressData: streetaddressData,
```

```
    cityData: cityData,
```

```
    zipcodeData: zipcodeData,
```

```
    country: country,
```

```
    phoneNumberData: phoneNumberData,
```

```
    passwordData: bcrypt.hashSync(passwordData, salt),
```

```
  };
```

```
  console.log("new_user", new_user);
```

```

const userCollection = await users();
const insertInfo = await userCollection.insertOne(new_user);
if (!insertInfo.insertedId) throw "Could not add user";

const inserted_user = await getUserById(insertInfo.insertedId);
console.log("inserted_user", inserted_user);
return { inserted_user: true };
} else {
  return { user_exists: true };
}

return { error: true };
};

const checkUser = async (username, password) => {
  //if user finds in db, return true
  username = username.toLowerCase();

  if (username.length < 4) {
    return {
      validation_error: "The length of the Username must be greater than 4 ",
    };
  }
  if (hasWhiteSpace(username)) {
    return {
      validation_error: "Username must not contain space character",
    };
  }
  if (hasWhiteSpace(password)) {

```

```

    return {
      validation_error: "Password must not contain space character",
    };
  }
  if (password.length < 6) {
    return {
      validation_error: "The length of the Password must be greater than 6 ",
    };
  }

  var userCheck = await getUserByUsername(username);
  if (userCheck == null) {
    return {
      validation_error: "Either the username or password is invalid",
    };
  } else {
    //verify password
    var pwdValid = bcrypt.compareSync(password, userCheck.passwordData); // true
    if (pwdValid == true) {
      //login success
      return { authenticatedUser: true, data: userCheck };
    } else {
      //invalid password
      return {
        validation_error: "Either the username or password is invalid",
      };
    }
  }
};

```

```

const checkEmail = async (email) => {
  var emailCheck = await getUserByEmail(email);
  if (emailCheck == null) {
    return {
      validation_error: "Email Id already exists, try with another email",
    };
  }
};

const checkPhone = async (phone) => {
  var phoneCheck = await getUserByPhone(phone);
  console.log("checkPhone", checkPhone);
  if (phoneCheck == null) {
    return {
      validation_error:
        "Phone Number already exists, try with another phone number",
    };
  }
};

//email adrees check from db
const getUserByEmail = async (email) => {
  const userCollection = await users();
  const userEmail = await userCollection.findOne({ emailData: email });
  console.log("getUserByEmail", userEmail);

  return userEmail;
};

//phonenumber check from db
const getUserByPhone = async (phone) => {

```

```

const userCollection = await users();
const userPhone = await userCollection.findOne({ phonenumberData: phone });
console.log("getUserByPhone", userPhone);

return userPhone;
};

const getUserByUsername = async (username) => {
  console.log("searching for " + username);
  const userCollection = await users();
  const user = await userCollection.findOne({ usernameData: username });
  console.log("getUserByUsername", user);

  return user;
};

const getUserById = async (id) => {
  const userCollection = await users();
  const userr_id = await userCollection.findOne({ _id: ObjectId(id) });
  return userr_id;
};

const updateRecord = async (
  id,
  firstnameData,
  lastnameData,
  dobData,
  genderData,
  streetaddressData,
  cityData,

```

```

    zipcodeData,
    country
  ) => {
    if (firstnameData.length <= 0) {
      return {
        validation_error: "The length of the first must be greater than 0! ",
      };
    }
    if (lastnameData.length < 4) {
      return {
        validation_error: "The length of the lastname must be greater than 0!",
      };
    }
    // //verify user does not exist in db
    // doesUserExist = await getUserByUsername(usernameData);

    // if (doesUserExist == null) {
    //user does not exist, therefore save record
    let updateUser;
    updateUser = {
      firstnameData: firstnameData,
      lastnameData: lastnameData,
      dobData: dobData,
      genderData: genderData,
      streetaddressData: streetaddressData,
      cityData: cityData,
      zipcodeData: zipcodeData,
      country: country,
    };
    console.log("update_user", updateUser);
  }
}

```

```

// console.log("_id", _id);
console.log("id", id);

const userCollection = await users();

// { _id: ObjectId(id) },
// { _id: id },
const updatedInfo = await userCollection.updateOne(
  { _id: ObjectId(id) },
  { $set: updateUser }
);

if (updatedInfo.modifiedCount === 0) {
  // throw "could not update user successfully";
  return {
    validation_error: "could not update user successfully",
  };
}

var userCheck = await getUserById(id);
//login success
return { updatedInfo: true, data: userCheck };

// return { error: true };

// updateEnrolledId
};

const updateEnrolledId = async (user_id, enroll_id) => {

```

```

let enrolledCourse = [];

// console.log("enrolledCourse", enrolledCourse);

// console.log("_id", _id);
console.log("user_id", user_id);

// let courseList = await coursess.getCourseById(enroll_id);

const userCollection = await users();

const updatedInfo = await userCollection.updateOne(
  { _id: ObjectId(user_id) },
  { $push: { enrolledCourse: enroll_id } }
);

if (updatedInfo.modifiedCount === 0) {
  // throw "could not update user successfully";
  return {
    validation_error: "failed to enroll the course ",
  };
}

var userCheck = await getUserById(user_id);
//login success
return { updatedInfo: true, data: userCheck };

// return { error: true };

// updateEnrolledId

```

```

};

//get all user data
const getAllUsers = async () => {
  const userCollection = await users();
  const userList = await userCollection.find({}).toArray();
  if (!userList) throw "No users in system!";

  //var userCheck = await getUserByUsername();

  // return { courseCollection: true, data: courseList };
  return userList;
};

const unenroll_course = async (userId, courseId) => {
  // let enrolledCourse = [];

  // console.log("user_id", enrolled_courses_hee);

  const userCollection = await users();

  let { _id, enrolledCourse } = await getUserByUsername(userId);

  console.log("USERRRRRRR-----", enrolledCourse);

  // if (enrolledCourse && enrolledCourse.length === 0) {
  //   return {
  //     validation_error: "Course not found in enrolled courses",
  //   };
  // }

```

```
// if(enrolledCourse && !enrolledCourse?.includes(courseId)){
//   return {
//     validation_error: "Course not found in enrolled courses",
//   };
// }
```

```
enrolledCourse = enrolledCourse?.filter((course) => course !== courseId);
```

```
console.log("USERRRRRRR-----2");
```

```
const updatedInfo = await userCollection.updateOne(
  { _id: ObjectId(_id) },
  { $set: { enrolledCourse: enrolledCourse } }
);
```

```
if (updatedInfo.modifiedCount === 0) {
  return {
    validation_error: "failed to remove the course ",
  };
}
```

```
var userCheck = await getUserByUsername(userId);
//login success
return { updatedInfo: true, data: userCheck };
};
```

```
module.exports = {
  createUser,
  checkUser,
```

```
updateRecord,  
updateEnrolledId,  
getAllUsers,  
getUserByUsername,  
unenroll_course,  
};
```

Додаток Б

Програмний код для дизайну веб-додатку з використанням HTML та CSS

Текст програми

Аркушів 15

Початкова сторінка користувача:

```
<main>
```

```
<link rel="stylesheet" href="/public/css/course_card.css" />
```

```
<div class="info">
```

```
<br />
```

```
<div class="card">
```

```
<div class="card-header">Ласкаво просимо до веб-сайту онлайн-навчання  
&#127891;</div>
```

```
<div class="card-body">
```

```
Цей вебсайт - це платформа для навчання.
```

```
Де викладач може створювати нові курси, студенти можуть реєструватися  
на курси,
```

```
можуть складати тестові іспити на курсі та завантажувати  
конспекти.</div>
```

```
</div>
```

```
</div>
```

```
<span style="color: red;"> {{error_msg}} </span>
```

```
<div class="card">
```

```
<div class="card-header">Сторінка входу: введіть ім'я користувача та пароль
```

```
</div>
```

```
<div class="card-body">
```

```
{ {! user login form } }
```

```
<br />
```

```

<form id="login-form" autocomplete="off" method="post" action="/login">
  <label for="usernameInput">Ім'я/Username </label>
  &#128100;
  <input
    id="usernameInput"
    type="text"
    name="usernameInput"
    placeholder="Username"
  />

```

```

<br /><br />
<label for="password">Пароль/Password </label>
&#128274;
<input
  id="passwordInput"
  type="password"
  name="passwordInput"
  placeholder="Password"
/>
<br /><br />

```

```

<button style="height:60px;" class="btn btn-outline">
  <a href="/instructor">Натисніть тут, щоб увійти в систему як викладач
&#128072;</a></button>
<button
  style="height:60px;width:90px;"
  class="btn"
  id="login-button"
  type="submit"
>Login</button>

```

```
<br />
```

```
</form>
```

```
</div>
```

```
<div class="card-footer">
```

```
  <button style="height:60px;" class="btn btn-outline"><a
    href="/register"
```

```
    >Натисніть тут, щоб зареєструвати користувача &#128072;</a></button>
```

```
  <button style="height:60px;" class="btn btn-outline"><a
    href="/instructor/instructor_register"
```

```
    >Натисніть тут, щоб зареєструвати викладача &#128072;</a></button>
```

```
</div>
```

```
</div>
```

```
{{#if error}}
```

```
  <h3>{{error}}</h3>
```

```
{{/if}}
```

```
</main>
```

Сторінка реєстрації користувача:

```
<main>
```

```
<h1>Сторінка реєстрації користувача</h1>
```

```
<span style="color: red;"> {{error_msg}} </span>
```

```
<div class="inputFieldsId" id="inputFieldsId">
```

```

<form
  id="registartion-form"
  autocomplete="on"
  method="post"
  action="/register"
>
  <br />
  {{!-- first name --}}
  <div>
    <label for="firstname">Ім'я/First Name* :</label>
    <input type="text" id="firstnameInput" name="firstnameInput"
autocomplete="name" enterkeyhint="next">
    {{!-- last name --}}
    <label for="lastname">Прізвище/Last Name* :</label>
    <input type="text" id="lastnameInput" name="lastnameInput"
autocomplete="name" enterkeyhint="next">
  </div>
  <br />

  {{! enter username }}
  <div>
    <label for="username">Нікнейм користувача/Enter Username* :</label>
    <input
      id="usernameInput"
      type="text"
      name="usernameInput"
      placeholder=""
    /></div>
  <br />

```

```

    {{! enter mail id }}

    <div>

    <label for="email">Введіть електрону пошту/Enter your email* :</label>

    <input type="email" id="emailInput" name="emailInput" placeholder="" />

</div>

    <br />

    {{! enter dob }}

    <div>    <label for="birthday">Дата народження/Birthday:</label>

    <input type="date" id="birthdayInput" name="birthdayInput" />

    </div>

<br />

    {{! choose gender }}

    <div>

    <label>Оберіть стать/Choose your gender:

    <select name="genderInput" id="genderInput">

    <option value="Male">Чоловік</option>

    <option value="Female">Жінка</option>

    </select>

    </label>

</div>

    <br />

    {{!-- address fields --}}

    {{> page/address}}

    {{! contact }}

```

```

<div>
  <label for="phone">Введіть номер телефону/Enter a phone number:</label>
  <input
    type="text"
    id="phoneInput"
    name="phoneInput"
    placeholder="123-45-678"

  /></div><br />
  { {! password } }

```

```

<div>
  <label for="password">Створіть пароль/Create Password* :</label>
  <input
    id="passwordInput"
    type="password"
    name="passwordInput"
    placeholder="Password"
  /></div>
  <br />

```

```

<br />

```

```

  <a href="/instructor/instructor_register">Натисніть тут, щоб переглянути
сторінку реєстрації викладача &#128072;</a><br /><br />

```

```

<div>
  <input id="submit-button" type="submit" value="Submit" class="mysubmit" />
</div><br/>

```

```
</form>
```

```
</div>
```

```
{{#if error}}
```

```
<h6>{{error}}</h6>
```

```
{{/if}}
```

```
</main>
```

```
<a href="/" id="hrefid"> <br />Вже є аккаунт? Натисніть тут, щоб увійти  
&#128072;</a><br />
```

```
<a href="/instructor/" id="hrefid"> <br />Вже є аккаунт? Натисніть тут, щоб  
увійти як викладач &#128072;</a> <br /><br />
```

Сторінка входу користувача:

```
<main>
```

```
<link rel="stylesheet" href="/public/css/course_card.css" />
```

```
<div class="info">
```

```
<br />
```

```
<div class="card">
```

```
<div class="card-header">Ласкаво просимо до веб-сайту онлайн-навчання  
&#127891;</div>
```

```
<div class="card-body">
```

```
Цей вебсайт - це платформа для навчання.
```

Де викладач може створювати нові курси, студенти можуть реєструватися на курси,

можуть складати тестові іспити на курсі та завантажувати конспекти.</div>

</div>

</div>

 {{error_msg}}

<div class="card">

<div class="card-header">Сторінка входу: введіть ім'я користувача та пароль

</div>

<div class="card-body">

{{! user login form }}

<form id="login-form" autocomplete="off" method="post" action="/login">

<label for="usernameInput">Ім'я/Username </label>

👤

<input

id="usernameInput"

type="text"

name="usernameInput"

placeholder="Username"

/>

<label for="password">Пароль/Password </label>

🔒

<input

id="passwordInput"

type="password"

name="passwordInput"

placeholder="Password"

/>

```
<br /><br />
```

```
<button style="height:60px;" class="btn btn-outline">
```

```
  <a href="/instructor">Натисніть тут, щоб увійти в систему як викладач  
&#128072;</a></button>
```

```
<button
```

```
  style="height:60px;width:90px;"
```

```
  class="btn"
```

```
  id="login-button"
```

```
  type="submit"
```

```
>Login</button>
```

```
<br />
```

```
</form>
```

```
</div>
```

```
<div class="card-footer">
```

```
  <button style="height:60px;" class="btn btn-outline"><a
```

```
    href="/register"
```

```
  >Натисніть тут, щоб зареєструвати користувача &#128072;</a></button>
```

```
  <button style="height:60px;" class="btn btn-outline"><a
```

```
    href="/instructor/instructor_register"
```

```
  >Натисніть тут, щоб зареєструвати викладача &#128072;</a></button>
```

```
</div>
```

```
</div>
```

```
{{#if error}}
```

```
  <h3>{{error}}</h3>
```

```
{{/if}}
```

```
</main>
```

Сторінка редагування інформації про користувача:

```

{{> page/head_nav}}
{{!-- {{> page/sidenavigation}} --}}
{{#if userType}}
    {{> instructor_page/sidenavigation}}
{{else}}
    {{> page/sidenavigation}}
{{/if}}
<div class="main_content">
    <div class="header">Редагувати сторінку користувача</div>

    <div class="info">
{{!-- <h4>Welcome</h4> --}}
<form
    id="registartion-form"
    autocomplete="on"
    method="post"
    action="/updateUser"
>
<h2><label>Ім'я: </label><input type="text" id="firstnameInput"
name="firstnameInput" value="{{fname}}" /></h2>
<h2><label>Прізвище:</label><input type="text" id="lastnameInput"
name="lastnameInput" value="{{lname}}" /></h2>
<h2><label>ID Користувача:</label>{{user}}</h2>
<h2><label>ID Електронної пошти:</label>{{email}}</h2>
<h2><label>Дата Народження :</label><input type="date" id="dobInput"
name="dobInput" value="{{dob}}" /></h2>

```

```

    {{!--      <label>Gender      :</label><input      type="text"      id="genderInput"
name="genderInput" value="{{gender}}"/> --}}

    <h2><legend>Оберіть стать:

        <select name="genderInput" id="genderInput">

            <option value="Male">Чоловік</option>

            <option value="Female">Жінка</option>

        </select>

    </legend></h2>

    <h2><label>Адреса      :</label><input      type="text"      id="addressInput"
name="addressInput" value="{{saddress}}"/></h2>

    <h2><label>Місто :</label><input type="text" id="cityInput" name="cityInput"
value="{{city}}"/></h2>

    <h2><label>Поштовий Індекс :</label><input type="text" id="zipcodeInput"
name="zipCodeInput" value="{{zipcode}}"/></h2>

    <h2><label>Місто      :</label><input      type="text"      id="countryInput"
name="countryInput" value="{{country}}"/></h2>

    <h2><label>Номер мобільного телефону:</label>{{phonenumbers}}</h2>

    <h2>Current Date/Time : {{date_time}}</h2>

    {{!-- logout --}}

    {{!-- <a href="/logout">Logout</a> --}}

    <h2><input id="submit-button" type="submit" value="Save" />

    <label><a href="/profile"> Cancel </a></h2></label></h2>

</form>

</div>

</div>

{{> page/footer}}

```

Сторінка реєстрації на курс:

```

{{> page/head_nav}}
{{!-- {{> page/sidenavigation}} --}}
{{#if userType}}
    {{> instructor_page/sidenavigation}}
{{else}}
    {{> page/sidenavigation}}
{{/if}}

```

```

<link rel="stylesheet" href="/public/css/course_card.css" />

```

```

<div class="main_content">

```

```

    <div class="header">Вітаємо, {{fname}}. Запишіться на доступні
курси</div>

```

```

    <div class="info">

```

```

        {{!-- <div>Welcome to the new "Online Course Website". This will be your
platform to all the tutorials we offer and your learning progress.

```

```

        If you are new to online course, we recommend you to start with our
<b>html</b> tutorial.

```

```

    </div> --}}

```

```

<div>

```

```

    {{#each courses}}

```

```

    <div class="card">

```

```

<div class="card-header">Назва Курсу : {{this.courseTitle}}
({{this.courseId}})</div>

<div class="card-body">Опис Курсу : {{this.courseDescription}}</div>

<div class="card-footer">
  <form id="login-form" autocomplete="off" method="post"
action="/enroll_course">
    <input id="enrollInputId" name="enrollInputId" type="hidden" aria-
setsize="10px" class="btn" value="{{this._id}}"/>
    <button style="height:60px;width:200px;" class="btn" id="login-button"
type="submit">

      {{#if already_enrolled}}
        Перейти до вивчення курсу
      {{else}}
        Зареєструватися
      {{/if}}
    </button>
  </form>

</div>
</div>
{{/each}}
</div>
</div>
</div>

```

Панель навігації користувача:

```

{{> page/head_nav}}
{{!-- {{> page/sidenavigation}} --}}
{{#if userType}}
    {{> instructor_page/sidenavigation}}
{{else}}
    {{> page/sidenavigation}}
{{/if}}
<div class="main_content">
    <div class="header">Сторінка панелі пошуку користувачів</div>
    <div class="info">
<h1> Сторінка панелі пошуку користувачів</h1>
</div>
    </div>

{{> page/footer}}

```

Сторінка облікових записів для зв'язку:

```

{{> page/head_nav}}
{{#if userType}}
    {{> instructor_page/sidenavigation}}
{{else}}
    {{> page/sidenavigation}}
{{/if}}

<div class="main_content">
    <div class="header">Облікові записи профілів у соціальних мережах</div>

```

```
<div class="info">
```

```
<div class="social_media">
```

```
<a href="https://www.facebook.com/" target="_blank">Слідкувати за нами на  
Facebook <i class="fab fa-facebook-f"></i></a><br/>
```

```
<a href="https://youtube.com" target="_blank">Слідкувати за нами в Instagram <i  
class="fab fa-youtube"></i></a><br/>
```

```
<a href="https://www.instagram.com/" target="_blank">Слідкувати за нами на  
Youtube <i class="fab fa-instagram"></i></a>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
{{> page/footer}}
```

Вигляд профілю користувача:

```
{{> page/head_nav}}
```

```
{{!-- {{> page/sidenavigation}} --}}
```

```
{{#if userType}}
```

```
  {{> instructor_page/sidenavigation}}
```

```
{{else}}
```

```
  {{> page/sidenavigation}}
```

```
{{/if}}
```

```
<div class="main_content">
```

```
<div class="header">Керування профілем користувача :
```

```
<a href="/editUser">Редагувати користувача <i class="fa fa-pen"></i> </a>  
</div>
```

```
<div class="info">
```

```

{{!-- <h4>Welcome</h4> --}}
<h2>Ім'я: {{fname}}</h2>
<h2>Призвіще: {{lname}}</h2>
<h2>Нікнейм: {{user}}</h2>
<h2>Електронна Пошта: {{email}}</h2>
<h2>Дата Народження: {{dob}}</h2>
<h2>Стать: {{gender}}</h2>
<h2>Адреса: {{saddress}}</h2>
<h2>Місто: {{city}}</h2>
<h2>Поштовий Індекс: {{zipcode}}</h2>
<h2>Місто: {{country}}</h2>
<h2>Номер Телефону: {{phonenumber}}</h2>

<h2>Current Date/Time : {{date_time}}</h2>

{{!-- logout --}}
{{!-- <a href="/logout">Logout</a> --}}

</div>
</div>
{{> page/footer}}

```

Додаток В

Програмний код для створення викладачем курсів

Текст програми

Аркушів 4

Код для роботи створення курсів:

```

const express = require("express");
const router = express.Router();
const instructors = require("../data/instructor");
const feedbacks = require("../data/feedback");
const coursess = require("../data/courses");

router.route("/createCourse1").get(async (req, res) => {
  //code here for GET
  const id_data = req.session.idData;
  try {
    let courseList = await coursess.getAllCourses();
    res.json(courseList);
  } catch (e) {
    res.sendStatus(500);
  }
});

// route to my create course
router
  .route("/createCourse")
  .get(async (req, res) => {
    //code here for GET
    let name = req.session.fname + req.session.lname;
    if (req.session.user) {
      res.render("instructor/createCourse", {
        userType: req.session.userType,
        title: "Create Course Page",

```

```

    name,
  });
} else {
  res.render("instructor/instructorLogin", { title: "instructor Login" });
}
})
.post(async (req, res) => {
  const id_data = req.session.idData;
  //fname+lname
  const course_name = req.body.courseNameInput;
  //select country
  const description_name = req.body.descriptionInput;
  //message
  const courseid_name = req.body.courseIdInput;
  const resource_name = req.body.courseInput;

  try {
    //Create a User by sending u and p.
    var registration_response = await coursess.createCourse(
      id_data,
      course_name,
      courseid_name,
      description_name,
      resource_name
    );

    //course-data
    // req.session.courseTitle = registration_response.data.courseTitle;
    // req.session.courseId = registration_response.data.courseId;
    // req.session.courseDescription =

```

```

// registration_response.data.courseDescription;
// req.session.courseResource = registration_response.data.courseResource;

// console.log("registration_response", registration_response);

// if ("inserted_user" in registration_response) {
//   res.redirect("/");
// } else {
//   res.status(500);
//   res.render("instructor/instructor_private", {
//     title: "register",
//     error_msg: "Internal Server Error",
//   });
// }
// } catch (e) {
//   console.log(e);
// }
// });

if ("inserted_user" in registration_response) {
  res.redirect("/instructor/");
} else if ("user_exists" in registration_response) {
  res.status(400);
  res.render("instructor/createCourse", {
    userType: req.session.userType,
    title: "instructor course",
    error_msg: " Course Already Exist",
  });
} else if ("validation_error" in registration_response) {
  res.status(400);
  res.render("instructor/createCourse", {

```

```
    userType: req.session.userType,  
    title: "instructor course",  
    error_msg: registration_response.validation_error,  
  });  
  } else {  
    res.status(500);  
    res.render("instructor/createCourse", {  
      userType: req.session.userType,  
      title: "instructor course",  
      error_msg: "Internal Server Error",  
    });  
  }  
  } catch (e) {  
    console.log(e);  
  }  
});  
module.exports = router;
```