Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Бородіна Павла Євгеновича*
(ПІБ)

академічної групи *122-19-3*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка клієнтської частини веб-застосунку для книжкового сервісу засобами Angular*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Гуліна І.Г.*			
розділів:				
спеціальний	*доц. Гуліна І.Г.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-3* *Бородіна Павла Євгеновича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка клієнтської частини веб-застосунку для книжкового сервісу засобами Angular*

затверджена наказом ректора НТУ «ДП» від «16» травня 2023 р. № 350-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Гуліна І.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Бородін П. Є.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

ЗМІСТ

РЕФЕРАТ

Пояснювальна записка: 75 с., 19 рис., 3 дод., 20 джерел.

Об’єкт дослідження: веб-застосунок для книжкового сервісу.

Мета кваліфікаційної роботи: розробка веб-застосунку “BooksAI” для надання можливості кінцевому користувачу (любителю книг) підбирати книги, схожі на улюблені.

У вступі здійснюється опис мети кваліфікаційної роботи та галузі, в якій вона застосовується, обгрунтовується актуальність її теми та проводиться уточнення постановки завдання.

В ході написання першого розділу проаналізовано предметну галузь, визначено призначення розроблюваної роботи та актуальність завдання, створена постановка завдання, описано вимоги до програмної імплементації, до використовуваних технологій та програмних засобів.

В ході написання другого розділу здійснено аналіз актуальних рішень, визначено платформу для розробки, здійснено проектування та розробку веб-застосунку, проведено опис алгоритму та структури функціонування веб-застосунку, визначені вхідні та вихідні дані, вказані характеристики параметрів технічних засобів, описано виклик та завантаження проекту, описана робота веб-застосунку.

В ході написання економічного розділу трудомісткість розробленого інформаційного веб-застосунку, здійснено підрахунок вартості роботи по створенню веб-застосунку та розраховано час його розробки.

Актуальність веб-застосунку визначається збільшенням попиту на послуги, що надаються сервісами штучного інтелекту, що скорочує час пошуку цікавих книг та значно спрощує і оптимізує обслуговування клієнтів.

Список ключових слів: ВЕБ-ЗАСТОСУНОК, ВЕБ-ДОДАТОК, БРАУЗЕР, ШТУЧНИЙ ІНТЕЛЕКТ.

ABSTRACT

Explanatory note: 75 sheets, 19 images, 3 app., 20 sources.

Object of study: web application for book service.

The purpose of the qualification work: development of a web application "BooksAI" to enable the end user (book lover) to select books similar to their favorites.

The introduction describes the purpose of the qualification work and the industry in which it is used, the relevance of its topic is justified and the task is clarified.

During the writing of the first section, the subject industry is analyzed, the purpose of the developed work and the relevance of the task were determined, the task is created, the requirements for software implementation were described, to the technologies and software used.

During the writing of the second chapter, an analysis of current solutions was carried out, a development platform was determined, the design and development of the web application was carried out, a description of the algorithm and structure of the functioning of the web application was carried out, the input and output data were determined, the characteristics of the parameters of the technical means were specified, the call and download of the project were described , the operation of the web application is described.

In the course of writing the economic section, the labor intensity of the developed information web application was calculated, the cost of work on creating a web application was calculated, and the time of its development was calculated.

The relevance of the web application is determined by the increase in demand for services provided by artificial intelligence services, which shortens the time of searching for interesting books and significantly simplifies and optimizes customer service.

Keywords: WEBSITE, WEB APPLICATION, BROWSER, ARTIFICIAL INTELLIGENCE.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IT – Information Technology;

HTML – Hyper Text Markup Language;

CSS – Cascading Style Sheets;

JS – JavaScript;

SCSS – Syntactically Awesome Style Sheet;

AJAX – Asynchronous JavaScript And XML;

DOM – Document Object Model;

ПК – Персональний Компьютер;

URL – Uniform Resource Locator;

HTTPS – Hypertext Transfer Protocol Secure;

AI – Artificial intelligence;

RxJS – Reactive Extensions for JavaScript;

JSON – JavaScript Object Notation;

SPA – Single-page application;

OOP – Object-oriented programming;

API – Application Programming Interface;

MVC – Model-View-Controller;

MVVM – Model-View-ViewModel;

CLI – Command-line interface;

REST – Representational State Transfer.

ВСТУП

У сучасному світі з кожним роком зростає комерційна популярність книжок. Зростає як кількість читачів, так і кількість видавців, адже доступність книжок стає все більшою, отже, збільшується конкуренція на ринку книжкових послуг. Необхідним атрибутом кожного читача у наш час є веб-застосунок, аби кожному читачу було зручніше вести облік книжок, які він прочитав, та рекомендацій по цім книжками від провідного AI-сервісу у тому самому веб-застосунку. Сучасні обчислювальні засоби та інформаційні технології (ІТ) здатні значною мірою автоматизувати роботу з обліку книжок та рекомендацій.

Використання обчислювальної техніки істотно спрощує формування рекомендацій по улюбленим книжкам та облік уже прочитаних книжок.

Метою даної кваліфікаційної роботи бакалавра є розробка інформаційного веб-додатку для читачів під назвою "BooksAI". Користувачем системи є читач. Реалізована у процесі роботи система має допомогти видавцю відстежувати свої прочитані книжки та рекомендації схожих книжок в одному просторі, що полегшить пошук книжок для подальшого читання.

Завдання проекту та об'єкт його діяльності безпосередньо пов'язані з освітньою програмою "Комп'ютерні науки" і відповідають загальній тематиці проектів у цій галузі.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

У підрозділі "Загальні відомості з предметної галузі" розглядається сучасний стан проблеми, пов'язаної з онлайн книжними сервісами, які забезпечують зберігання відгуків на книжки та надають рекомендації користувачам. Аналізується наявність аналогічних рішень, рівень вирішення завдань, існуючі технічні протиріччя, прогалини в знаннях у цій галузі, а також нездійснені вимоги до виробів або розробок наукового, організаційного або іншого характеру.

Онлайн книжні сервіси для зберігання відгуків та підбору рекомендацій надають користувачам можливість відзначати та залишати відгуки на прочитані книжки, а також отримувати персоналізовані рекомендації щодо подальшого читання. Вони використовуються у веб-сайтах та мобільних додатках, які дозволяють користувачам взаємодіяти зі спільнотою книголюбів, обмінюватися думками про книги та знаходити нові цікаві твори.

Характеристика галузі застосування програми або програмного виробу:

- Зберігання відгуків на книжки: Онлайн книжні сервіси дозволяють користувачам залишати відгуки, оцінки та коментарі на книжки, які вони прочитали. Це створює можливість формування бази даних з відгуками, які можуть бути використані для оцінки книг та підбору рекомендацій.
- Підбір рекомендацій: На основі зібраних відгуків та інформації про вподобання користувачів, онлайн книжні сервіси надають персоналізовані рекомендації щодо подальшого читання. Застосовуються алгоритми машинного навчання та фільтри, які

аналізують історію читання та вподобання користувача для визначення книжок, які можуть бути йому цікаві.

- Спільнота книголюбів: Онлайн книжні сервіси створюють веб-сайти та додатки, які об'єднують книголюбів у спільноту. Користувачі можуть обговорювати книги, обмінюватися рекомендаціями та відгуками, вступати до читацьких груп та брати участь у літературних подіях.

Об'єктом, в якому використовуються онлайн книжні сервіси для зберігання відгуків та підбору рекомендацій, є електронні книжки. Користувачі можуть залишати свої відгуки та оцінки на конкретні книги, а отримані рекомендації спрямовуються на підбір нових книг для читання відповідно до індивідуальних інтересів та вподобань.

1.2. Призначення розробки та галузь застосування

Система, або програма, має назву «BooksAI».

Основна термінологія, пов'язана з цією розробкою, включає такі ключові слова як «книжний сервіс», «книжні полиці», «пошук схожих книжок» та «OpenAI».

Причиною виникнення необхідності розробки цього програмного забезпечення є збільшення популярності онлайн книжних сервісів та потреба користувачів у зручному способі організації та пошуку книжок. Книжний сервіс, що розробляється, має на меті надати користувачам можливість створювати віртуальні полиці для організації своєї особистої колекції книжок, а також здійснювати пошук схожих книжок на основі запитів, використовуючи функціональність OpenAI.

Галузь застосування цієї системи охоплює онлайн книжний ринок та спільноту книголюбів. Користувачі, що шукають зручний спосіб організації своєї електронної книжкової колекції, зможуть скористатися цим сервісом для

створення віртуальних полиць, розміщення на них книжок та отримання персоналізованих рекомендацій щодо подібних книжок. Також система може бути корисною для книжних магазинів та видавництв, які шукають способи покращити пошук та рекомендації книжок для своїх клієнтів.

1.3. Підстава для розробки

Підставами для виконання кваліфікаційної роботи є:

- ОПП за спеціальністю 122 «Комп'ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;
- завдання на кваліфікаційну роботу на тему «Розробка клієнтської частини веб-застосунку для книжкового сервісу засобами Angular».

1.4. Постановка завдання

В даній кваліфікаційній роботі розглядається створення інформаційної веб-орієнтованої системи для впровадження інноваційного книжкового сервісу, який буде проводоти маніпуляції з книжками на книжкових полицях (створення полиць, підбір подібних книжок за наявними тощо) за допомогою провідного AI-сервіса «OpenAI».

Інформаційна система створюється для надання можливості робити підбір книжок, подібних на улюблені. Аудиторію інформаційної системи становлять люди у віці від 10 до 80 років.

Тематикою інформаційної системи є реалізація веб-частини книжкового додатка, у якому існує інтеграція с провідним AI-сервісом для автоматизації підбору подібних книжок.

Інформаційна система повинна задовольняти наступним основним вимогам: висока швидкість завантаження сторінок; сучасний мінімалістичний дизайн; максимальна зручність роботи із системою для користувача; оптимізація сторінок під пошукові системи; легкість сприйняття інформації; простота й повнота управління змістом.

Створення й розробка інформаційної системи повинна включати наступні етапи:

- Аналіз вимог: Проведення детального аналізу вимог до інформаційної системи, зокрема до книжного сервісу. Визначення, які функції має надавати система, які дані потрібно зберігати, які операції користувачі повинні мати змогу виконувати та інші важливі аспекти.
- Проектування бази даних: Враховуючи вимоги до системи, розробка структури бази даних, яка буде використовуватися для зберігання інформації про книжки та полиці. Використання Json Server для реалізації цієї бази даних.
- Розробка користувацького інтерфейсу: Використання Angular 15 та PrimeNg для розробки користувацького інтерфейсу веб-застосунку. Створення сторінки для формування книжкових полиць, пошуку схожих книжок та відображення результатів [4].
- Інтеграція з OpenAI API: Взаємодія з OpenAI API для отримання рекомендацій та інших даних на основі запитів користувачів. Наприклад, для створення нової книжкової полиці; використання OpenAI API для отримання рекомендацій щодо схожих книжок, які можуть бути додані до полиці [11].
- Реалізація функціональності: Реалізація основних функцій, таких як створення книжкових полиць, пошук схожих книжок, відображення результатів та інші операції, які були визначені на етапі аналізу вимог. Використання RxJs для керування асинхронними операціями та обробки даних [8].

- Тестування: Проведення тестування системи, щоб переконатися, що вона працює належним чином та відповідає вимогам. Виконання модульного тестування компонентів, інтеграційне тестування та випробування системи на різних сценаріях використання.
- Підтримка та обслуговування: Забезпечення підтримки користувачів та виправлення помилок або вдосконалення системи після її випуску. Реагування на зворотний зв'язок користувачів та вдосконалення системи згідно з новими потребами або вимогами.

Це загальний сценарій створення та розробки інформаційної системи для книжного сервісу, що використовує Angular 15, PrimeNg, RxJs та Json Server. Конкретні кроки та деталі можуть змінюватися в залежності від вимог користувачів.

Завданням кваліфікаційної роботи є розробка інформаційного сайту для читача. Ознайомлення користувача з наданою інформацією на сайті та демонстрація можливостей даного проекту.

Программа повинна імплементувати такі функції:

- просте та доступне користування сайтом читачами;
- відображення сторінок сайту на різних девайсах та браузерах;
- захист даних від роботів та зловмисників.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставленої в роботі мети в інформаційній системі, що розробляється, повинні бути реалізовані такі функціональні характеристики:

- можливість пошуку книжок;
- можливість створення полиці з книжками;
- можливість перегляду доданих книжок у книжковій полиці;
- можливість користувача отримати доступ до своїх полиць;

- можливість переглянути рекомендации AI-сервісу за своїми книжками;
- швидкість роботи з базою;
- зручний сучасний мінімалістичний дизайн.

1.5.2. Вимоги до інформаційної безпеки

Під інформаційною безпекою розуміють стан захищеності систем обробки і зберігання даних, при якому забезпечено конфіденційність, доступність і цілісність інформації, використання й розвиток в інтересах громадян або комплекс заходів, спрямованих на забезпечення захищеності інформації особи, суспільства і держави від несанкціонованого доступу, використання, оприлюднення, руйнування, внесення змін, ознайомлення, перевірки запису чи знищення.

Вимоги до інформаційної безпеки для книжного сервісу, який може створювати книжні полиці та шукати схожі книжки за наданними у полиці:

1. Загроза безпеці даних: Один з основних аспектів інформаційної безпеки полягає в забезпеченні конфіденційності, цілісності та доступності даних користувачів. У цьому випадку, важливо захистити інформацію про книжкові полиці, пошукові запити та персональні дані користувачів.

Шляхи усунення загроз:

- Використання криптографічних методів шифрування для захисту конфіденційності даних. Важливо застосовувати сильні алгоритми шифрування для зберігання та передачі даних.
- Реалізація механізмів автентифікації та авторизації, щоб забезпечити правильний доступ до інформації тільки авторизованим користувачам.

- Регулярне оновлення та патчі системи для запобігання використанню вразливостей, які можуть бути використані для несанкціонованого доступу до даних.

2. Загроза атак з боку зловмисників: Книжний сервіс може піддаватися різним видам кібератак, таким як SQL-ін'єкції, перехоплення сеансу, атаки на введення даних тощо.

Шляхи усунення загроз:

- Застосування механізмів валідації та санітизації введених даних для запобігання SQL-ін'єкціям та іншим видам атак на введення.
- Використання безпечних протоколів передачі даних, таких як HTTPS, для запобігання перехопленню сеансів та забезпечення конфіденційності даних [19].
- Регулярні аудити безпеки та тестування на проникнення, щоб виявляти потенційні слабкі місця системи та вживати заходів для їх усунення.

3. Загроза недоступності сервісу: Напади, такі як DDoS атаки, можуть спричинити недоступність книжного сервісу для користувачів.

Шляхи усунення загроз:

- Використання захисту від DDoS атак, таких як фільтрація трафіку, кешування та розподіл навантаження, для запобігання перевантаження серверів та забезпечення стабільності сервісу.
- Резервне копіювання та відновлення даних, щоб забезпечити можливість відновлення системи у разі недоступності або втрати даних.
- В цілому, для забезпечення безпеки книжного сервісу, важливо реалізувати надійні механізми шифрування, аутентифікації та авторизації, проводити регулярні аудити безпеки та тестування на

проникнення, а також мати механізми для запобігання та виявлення атак та недоступності сервісу.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для нормального функціонування веб-орієнтованої інформаційної системи, що використовує Angular 15, Json Server DB та OpenAI API, необхідно врахувати наступні мінімальні вимоги до клієнтської частини:

- Веб-переглядач: Користувачі повинні мати доступ до сучасного веб-переглядача, який підтримує HTML5, CSS3 та ECMAScript 6. Рекомендовані веб-переглядачі включають Google Chrome, Mozilla Firefox, Safari або Microsoft Edge.
- Операційна система: Веб-застосунок може працювати на різних операційних системах, таких як Windows, macOS або Linux. Варто переконатися, що система користувача підтримується та сумісна зі специфікаціями Angular 15.
- Апаратні вимоги: Мінімальні апаратні вимоги зазвичай залежать від операційної системи та веб-переглядача. Зазвичай, комп'ютери з середньою продуктивністю, такі як процесор Intel Core i3, 4 ГБ оперативної пам'яті та достатньою кількістю вільного місця на жорсткому диску, повинні бути достатніми для запуску веб-застосунку.
- Мережевий доступ: Для використання веб-застосунку користувачеві потрібний доступ до Інтернету. Він повинен мати стабільне з'єднання з достатньою швидкістю для завантаження веб-сторінок та взаємодії з API-сервісами, такими як Json Server DB та OpenAI API.

- Підтримка JavaScript: Веб-переглядач користувача повинен підтримувати виконання JavaScript-скриптів, оскільки Angular 15 та OpenAI API використовують JavaScript для функціонування [3].
- Передача даних: Деякі функції веб-застосунку можуть вимагати передачі даних через мережу. Тому важливо переконатися, що користувач має дозвіл на взаємодію з відповідними API та їх методами, які використовуються для отримання даних з Json Server DB та взаємодії з OpenAI API.
- Безпека: Забезпечення механізмів безпеки, таких як захист від CSRF-атак, обмеження прав доступу користувачів та шифрування даних, що передаються через мережу. Рекомендується використовувати найновіші версії Angular та стежити за оновленнями безпеки.

Це лише загальний огляд мінімальних вимог до клієнтської частини веб-застосунку, що використовує Angular 15, Json Server DB та OpenAI API. Залежно від конкретних потреб користувачів, можуть знадобитися додаткові налаштування та вимоги до середовища.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

1.5.4. Вимоги до інформаційної та програмної сумісності

Вимоги до інформаційної та програмної сумісності для розробленої системи «BooksAI» є наступними:

1. Інформаційні структури: Система повинна мати визначену структуру для збереження інформації про книжки, полиці та інші додаткові дані, необхідні для її функціонування. Використована база даних Json Server повинна відповідати цим вимогам і мати відповідну структуру для збереження даних.
2. Вихідні коди: Розроблені вихідні коди системи повинні бути написані з використанням Angular 15, PrimeNg, RxJs та інших використаних засобів. Коди повинні бути читабельними, добре організованими та дотримуватися кращих практик програмування.
3. Мова програмування: Для розробки системи використано мову програмування TypeScript, яка є однією з основних мов програмування для розробки Angular-додатків. Розроблені модулі, компоненти та сервіси повинні бути написані на TypeScript [7].
4. Програмні засоби: Для створення та функціонування системи використано різні програмні засоби, такі як:
   - Angular 15: Фреймворк для розробки веб-додатків, який надає структуру та інструменти для побудови клієнтської частини системи.
   - RxJs: Бібліотека для реактивного програмування, яка дозволяє ефективно керувати асинхронними операціями та потоками даних.
   - Json Server: Простий сервер для розробки, що дозволяє створювати та використовувати просту базу даних з використанням JSON-файлів [5].

Враховуючи ці вимоги до програмної сумісності, розробка системи «BooksAI» дозволить створити веб-застосунок, який може створювати книжні полиці та шукати схожі книжки на основі запитів до OpenAI API.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Основні функції цієї програми включають:

- Створення книжних полиць: Користувач може створювати власні книжні полиці, надаючи їм назву та додавши книжки до полиці з використанням інтерфейсу користувача.
- Додавання книжок до полиць: Користувач може додавати книжки до створених книжних полиць. Для цього він може вказати дані про книжку, такі як назва, автор, жанр тощо, або використати інтеграцію з OpenAI, щоб отримати рекомендації щодо схожих книжок.
- Пошук схожих книжок: Користувач може використовувати функцію пошуку, щоб знайти схожі книжки на основі книжок, що вже знаходяться на його полиці. Для цього використовується OpenAI API, який надає рекомендації та інші дані на основі запитів користувача.
- Відображення результатів: Результати пошуку схожих книжок відображаються користувачу у вигляді списку або іншої зручної форми. Користувач може переглядати деталі про книжки, включаючи назву, опис, автора та інші важливі відомості.

2.2. Опис застосованих математичних методів

У процесі проектування та розробки даної інформаційної системи не використовувалися складні математичні методи або алгоритми. Були

використані базові арифметичні операції та логічні вирази для обробки та маніпуляції даними. Математичні методи, такі як статистичний аналіз, алгоритми машинного навчання або штучний інтелект (в арифметичних цілях), не застосовувалися у рамках цього проекту.

2.3. Опис використаних технологій та мов програмування

У процесі виконання даної кваліфікаційної роботи було вирішено використовувати архітектурний стиль REST для забезпечення обміну даними між сервером та клієнтською частиною застосунку.

Архітектура REST (Representational State Transfer) є архітектурним стилем для проектування розподілених систем, який базується на принципах використання веб-протоколу HTTP. Нижче наведено опис архітектури REST за заданим сценарієм дипломного проекту, включаючи використані технології та мови програмування [16].

Клієнтська частина:

- Веб-браузер або інша програма, яка взаємодіє з веб-сервісом.
- Використання Angular 15 для розробки користувацького інтерфейсу та взаємодії з сервером.
- Використання PrimeNg для швидкого створення і стилізації компонентів інтерфейсу.

Серверна частина:

- Використання Json Server для створення простої бази даних на основі JSON-файлів.
- Використання RxJs для реактивного програмування та керування асинхронними операціями.
- Використання мови програмування TypeScript для розробки серверної логіки.

Комунікація між клієнтом та сервером:

- Використання протоколу HTTP для обміну даними між клієнтом та сервером.
- Використання RESTful API для створення ресурсів та виконання операцій над ними.
- Передача даних у форматі JSON для зручності обробки.

Архітектурні принципи REST:

- Клієнт-сервер: Існує чітке розмежування між клієнтом і сервером, де клієнт відправляє запити, а сервер надає відповіді.
- Безстанцевість: Кожен запит від клієнта містить усю необхідну інформацію для обробки запиту, сервер не зберігає стан клієнта.
- Кешування: Відповіді сервера можуть бути кешовані на клієнті або проксі-сервері для покращення швидкодії та зменшення навантаження на сервер.
- Єдинообразність інтерфейсу: Клієнт та сервер використовують уніфікований інтерфейс, який специфікується стандартами HTTP (GET, POST, PUT, DELETE тощо).

Ця архітектура REST забезпечує простоту, масштабованість та легкість розробки та розширення системи. Вона дозволяє клієнту взаємодіяти з сервером шляхом виконання HTTP-запитів до RESTful API та отримувати/оновлювати дані у форматі JSON.

Для реалізації даної кваліфікаційної роботи було використано вказані мови програмування, розмітки та опису стилів:

- TypeScript.
- HTML5.
- CSS (SASS).

TypeScript - це мова програмування, яка є надмножиною (superset) мови JavaScript. Вона пропонує додаткові можливості та функції, які полегшують розробку програмного забезпечення. TypeScript надає статичну типізацію, що

дозволяє визначати типи даних для змінних, параметрів функцій та інших елементів програми.

Основні особливості TypeScript:

- Статична типізація: TypeScript дозволяє визначати типи даних для змінних, функцій, параметрів та інших елементів програми. Це полегшує виявлення помилок під час компіляції і поліпшує роботу з інструментами розробки.
- Об'єктно-орієнтований підхід: TypeScript підтримує об'єктно-орієнтоване програмування, включаючи класи, спадкування, інтерфейси та інші концепції ООР. Це дозволяє розробникам створювати структуровані, модульні та легко зрозумілі програми.
- Розширення JavaScript: TypeScript є надмножиною JavaScript, що означає, що весь існуючий JavaScript-код може бути використаний у проекті TypeScript. Це дозволяє поступово впроваджувати TypeScript у вже існуючі проекти без необхідності переписування коду з нуля.
- Інструменти розробки: TypeScript має розширений набір інструментів розробки, таких як компілятор TypeScript, який перетворює код TypeScript у виконуваний JavaScript, і інтегровані середовища розробки (наприклад, Visual Studio Code), які надають підсвічування синтаксису, автодоповнення та інші корисні функції.

Українські бакалаврські програми часто включають TypeScript як частину курсу розробки веб-додатків або програмного забезпечення загалом. Використання TypeScript дозволяє створювати більш безпечні, структуровані та ефективні програми, що є важливими аспектами сучасної розробки програмного забезпечення.

HTML5 (HyperText Markup Language 5) - це остання версія стандарту мови розмітки гіпертексту, яка використовується для створення та

відображення веб-сторінок [1]. Вона є основним стандартом для створення веб-контенту та визначає структуру та вигляд веб-сторінок.

Основні особливості HTML5:

- Розширені можливості: HTML5 надає розширені можливості порівняно з попередніми версіями HTML. Він включає нові елементи, атрибути та API, які дозволяють розробникам створювати більш складні та інтерактивні веб-сторінки.
- Підтримка мультимедіа: HTML5 має вбудовану підтримку для відео та аудіо, що дозволяє вбудовувати мультимедійний контент без необхідності використання сторонніх плагінів, таких як Adobe Flash.
- Графічні можливості: HTML5 надає можливості для малювання графіки безпосередньо на веб-сторінці за допомогою елемента canvas. Це дозволяє створювати складні візуальні ефекти, графіки та ігри без використання сторонніх плагінів.
- Мобільна підтримка: HTML5 пропонує покращену підтримку для мобільних пристроїв, що дозволяє створювати веб-сторінки, що адаптуються до різних розмірів екранів та працюють оптимально на мобільних пристроях.
- Локальне сховище даних: HTML5 має можливості для збереження даних локально на пристрої користувача. Це дозволяє створювати веб-додатки, які можуть працювати без підключення до Інтернету.

HTML5 є фундаментальним елементом сучасної веб-розробки. Вивчення HTML5 дозволяє спеціалістам розробляти веб-сторінки та веб-додатки з урахуванням сучасних стандартів та рекомендацій.

CSS (Cascading Style Sheets) - це мова опису стилів, яка використовується для визначення зовнішнього вигляду та форматування веб-сторінок. Вона використовується разом з HTML для визначення, як елементи сторінки

повинні бути відображені на екрані, включаючи кольори, шрифти, розміри, відступи, позиціонування та інші стилістичні властивості.

SASS (Syntactically Awesome Style Sheets) - це розширення CSS, яке додає додаткові можливості та функціональність до CSS. SASS використовує синтаксис, що розширює можливості звичайного CSS, дозволяючи використовувати змінні, вкладені селектори, міксіни, функції та інші конструкції, які полегшують розробку та підтримку стилів [2].

Основні особливості CSS та SASS:

- Селектори: CSS використовує селектори для вибору елементів сторінки, до яких будуть застосовуватись стилі. Це може бути вибір за тегом, класом, ідентифікатором або іншими атрибутами. SASS дозволяє використовувати вкладені селектори, що полегшує структурування та організацію стилів.
- Властивості та значення: CSS містить широкий набір властивостей та значень, які дозволяють визначати різні аспекти стилізації елементів. Наприклад, це можуть бути кольори, шрифти, розміри, відступи, рамки та інші. SASS дозволяє використовувати змінні, що дозволяють задавати значення один раз та використовувати їх багаторазово у стилях.
- Міксіни та функції: SASS надає можливість використовувати міксіни, що дозволяють згрупувати певні стилі та використовувати їх повторно. Також можна використовувати функції для обчислення значень стилів.
- Імпорт та спадкування: SASS дозволяє імпортувати стилі з інших файлів, що полегшує організацію та управління стилями. Також SASS підтримує механізм спадкування, що дозволяє успадковувати стилі від батьківських елементів.

CSS та SASS є необхідними для розробки веб-інтерфейсів та стилізації веб-додатків у галузі інформаційних технологій. Вивчення цих технологій

дозволить спеціалістам ефективно та професійно оформлювати веб-сторінки з використанням розширених можливостей стилізації та організації стилів.

Також для реалізації проекту було використано такі фреймворки та бібліотеки:

- Angular.
- JSON Server.
- OpenAI.

Angular - це платформа та фреймворк для розробки веб-додатків. Він використовується для побудови ефективних та масштабованих односторінкових додатків (SPA) з багатофункціональним інтерфейсом користувача [9].

Основні особливості Angular:

- TypeScript: Angular розроблений з використанням мови програмування TypeScript, яка є розширенням JavaScript. TypeScript надає статичний типізацію, покращує розробку та підтримку коду, забезпечує можливість використання класів, модулів та інших конструкцій ООР.
- Компонентна архітектура: Angular побудований на основі компонентної архітектури, де кожен елемент інтерфейсу користувача є окремим компонентом. Це сприяє розділенню відповідальностей, повторному використанню коду та забезпечує чистоту коду.
- Директиви: Angular надає багато вбудованих директив, які дозволяють контролювати поведінку та вигляд елементів сторінки. Наприклад, директиви можуть використовуватись для управління видимістю, станом, анімаціями та іншими аспектами елементів.
- Сервіси та залежності: Angular підтримує використання сервісів для обробки бізнес-логіки та обміну даними між компонентами. Залежності між компонентами та сервісами керуються системою

ін'єкції залежностей (Dependency Injection), що спрощує управління та тестування додатків.

- Роутинг: Angular надає можливість налаштування маршрутів, що дозволяє переключатись між різними компонентами та відображати вміст в залежності від URL. Це дозволяє створювати багатосторінкові додатки зі зручною навігацією.

Angular є одним з найпопулярніших фреймворків для веб-розробки та широко використовується в сфері IT.

JSON Server - це простий сервер, який дозволяє швидко створювати та запускати прототипи веб-сервісів з використанням файлів JSON як джерела даних. Він надає можливість емулювати поведінку справжнього сервера, відповідаючи на HTTP-запити, такі як GET, POST, PUT, DELETE і т.д.

Основні особливості JSON Server:

- Простота налаштування: JSON Server легко встановлюється та налаштовується за допомогою конфігураційного файлу. Для створення API не потрібно писати складний серверний код або налаштовувати бази даних.
- Реалістичні дані: створення власних файлів JSON з даними або використання існуючих файлів JSON. Це дозволяє емулювати реальні дані, які будуть використовуватися додатком.
- Маршрутизація: JSON Server дозволяє визначати маршрути та поведінку для різних запитів. Ви можете вказати, як сервер повинен реагувати на конкретний URL та тип запиту.
- Підтримка запитів: JSON Server надає підтримку різних типів запитів, таких як GET, POST, PUT, DELETE. Це дозволяє взаємодіяти з даними через HTTP-запити.

JSON Server є корисним інструментом для швидкого прототипування та розробки веб-додатків, особливо коли вам потрібно швидко отримати простий API для використання в клієнтському додатку. В Україні JSON Server активно

використовується в освітніх проектах та прототипах, дозволяючи студентам та розробникам швидко створювати і тестувати веб-додатки з мінімальними зусиллями.

OpenAI Node.js API - це програмний інтерфейс (API), який надає доступ до мовної моделі GPT-3.5 створеної компанією OpenAI. Цей API дозволяє розробникам взаємодіяти з мовною моделлю GPT-3.5 та використовувати її для різних завдань обробки природної мови.

Основні особливості OpenAI Node.js API:

- Генерація тексту: За допомогою OpenAI Node.js API можна генерувати тексти на основі вхідного контексту або запиту. Мовна модель GPT-3.5 може створювати розумні відповіді, повідомлення, статті та багато іншого на основі наданого контексту.
- Відповіді на запити: використання OpenAI Node.js API для отримання відповідей на питання, створення повідомлень, доповнення речень та багато іншого. Мовна модель може аналізувати вхідний текст та надавати контекстуальні відповіді.
- Заповнення тексту: За допомогою API ви можете використовувати мовну модель для автоматичного заповнення текстових шаблонів. Наприклад, ви можете створити заголовки, описи, списки або інші частини тексту на основі певного контексту.
- Переклад тексту: OpenAI Node.js API підтримує можливість перекладу тексту з однієї мови на іншу. Ви можете використовувати мовну модель для здійснення перекладу речень, фраз або повних текстових блоків.

OpenAI Node.js API є потужним інструментом для використання мовної моделі GPT-3.5 в різних сферах, таких як генерація тексту, розумний аналіз, переклад тексту та багато іншого. Українські студенти та розробники можуть

використовувати цей API для розробки різноманітних мовних додатків та досліджень у галузі обробки природної мови.

2.4. Опис структури системи та алгоритмів її функціонування

Основні компоненти цього паттерну в Angular:

- Модель (Model): Модель представляє дані, що використовуються в додатку. В Angular модель може бути представлена класами TypeScript, які містять властивості та методи для роботи з даними.
- Вигляд (View): Вигляд відповідає за візуальне представлення даних. В Angular вигляд визначається з використанням HTML шаблонів, в які вставляються директиви Angular для розширення функціональності та обробки подій.
- Контролер або Компонент (Controller or Component): Контролер або компонент виконує логіку додатку і забезпечує зв'язок між моделлю та виглядом. В Angular компоненти використовуються для організації структури додатку, обробки подій, взаємодії з користувачем та виклику сервісів.

Крім цього, в Angular використовується Dependency Injection (DI), який допомагає врегулювати залежності між компонентами, сервісами та іншими об'єктами, спрощує тестування та забезпечує легку розширюваність додатку.

Загалом, використання архітектурного паттерну MVC або MVVM разом з принципами Angular дозволяє ефективно організувати та підтримувати структуру Angular додатку, розділяючи логіку, дані та візуалізацію [12].

Angular має декілька переваг для структурування проекту:

Компонентна архітектура: Angular базується на компонентній архітектурі, що дозволяє розбити додаток на невеликі, самодостатні та

повторно використовувані компоненти. Це сприяє чіткості і структурованості проекту, полегшує розробку, тестування та підтримку коду.

- Модульність: Angular дозволяє організувати додаток у модулі, що дозволяє розділити функціональність на логічні частини. Кожен модуль містить компоненти, сервіси та інші ресурси, пов'язані між собою. Це полегшує розподіл роботи між командою розробників та сприяє повторному використанню модулів в інших проектах.
- Строга типізація: Angular використовує мову TypeScript, яка надає статичну типізацію. Це допомагає виявляти помилки на етапі компіляції та полегшує рефакторинг коду. Строга типізація також поліпшує читабельність коду та сприяє більшій надійності програми.
- Розширені можливості шаблонізації: Angular надає потужні можливості для створення шаблонів візуалізації. Використовуючи шаблони Angular, можна швидко створювати динамічний інтерфейс користувача, забезпечувати зв'язування даних та виконувати різноманітні операції з відображенням.
- Ефективне управління станом: Angular пропонує механізми для ефективного управління станом додатку. Зокрема, він має вбудовану систему обробки подій, двустороннє зв'язування даних та підтримку реактивного програмування за допомогою RxJS. Це допомагає уникати змішування стану та логіки додатку, спрощує відлагодження та покращує продуктивність додатку.
- Розширюваність: Angular надає багато можливостей для розширення функціональності додатку. Це включає в себе створення власних директив, пайпів, сервісів та інших розширень. Можливість розширення дозволяє розробникам впроваджувати унікальну функціональність та пристосовувати Angular під свої потреби.

Загалом, використання Angular допомагає структурувати проект, полегшує розробку, підтримку та розширення додатку. Його компонентна архітектура, модульність, строга типізація, можливості шаблонізації та ефективне управління станом роблять Angular потужним інструментом для розробки веб-додатків.

Всі перераховані концепції допомогли створити добре структурованний веб-додаток.

Загальна структура проекту (рис 2.1):

- src/app – головна директорія проекту;
- api – директорія з компонентом, який відповідає за обмін даними с серверною частиною;
- core – директорія для ключових компонентів проекту, які можуть не відносяться до основного функціоналу проекту;
- features – директорія для основних компонентів проекту, які створюють візуальну частину додатку;
- models – директорія із сутностями, які описують структури основних об'єктів проекту;
- pipes – директорія з перетворювачами даних;
- services – директорія з компонентами, які відповідають за обробку даних;
- файли з приставкою app – центральний компонент проекту, який відповідає за відображення всього проекту.

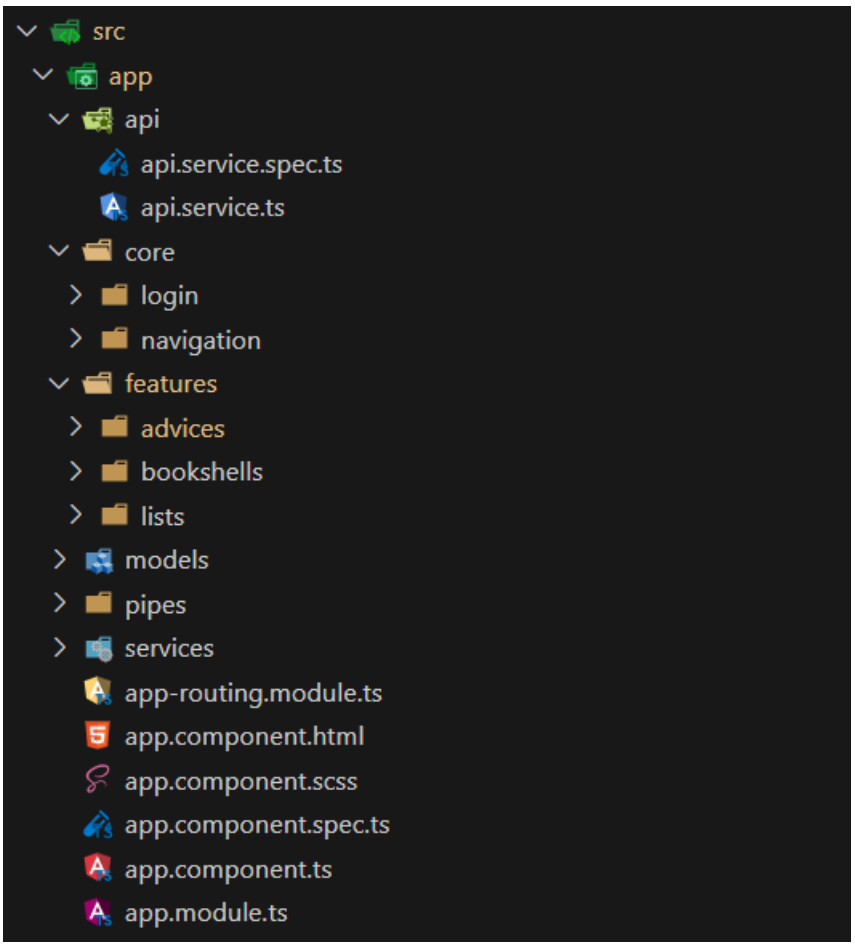

Рис. 2.1. Директорія компонентів проекту

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Обмін даними між сервером і клієнтським застосунком здійснюється за допомогою JSON, що є зручним форматом для обох сторін - як для обробки на сервері, так і для використання на клієнтській частині додатку та відповідає REST архітектурі.

Вхідні дані для цього проекту можуть бути різноманітні. Сервіси приймають такі дані, як книжні полиці та книжки.

Вихідні дані, які додаток надає кінцевому користовачеві такі:

- книжка з описом та автором, яку кінцевий користовач отримує через пошук;

- книжкова полиця, яку кінцевий користовач може створити сам;
- рекомендації від сервісу OpenAI, які базаються на книжках із книжкової полиці.

2.6. Опис розробленої системи

Розроблений додаток допоможе користувачу зберігати книжки та знаходити рекомендації по тим книжкам, які сподобавлися за допомогою провідного сервісу OpenAI. Сучасний мінімалістичний дизайн допоможе легше орієнтуватися в системі.

2.6.1. Використані технічні засоби

Під час розробки було використано ноутбук HP EliteBook G7 з такими характеристиками:

- операційна система: Windows 10 Pro;
- процесор: Intel Core i7-10th Gen;
- графічний адаптер: NVIDIA GeForce GTX;
- зберігання: SSD 512 ГБ;
- оперативна пам'ять: 16 ГБ DDR4.

Ці характеристики надають ноутбуку високу продуктивність та потужність для розробки веб-додатків для платформи Windows.

2.6.2. Використані програмні засоби

В процесі розробки проекту були використані наступні програмні засоби:

- Visual Studio Code (VSCode) - це інтегроване середовище розробки, яке надає широкі можливості для написання,

відлагодження та керування кодом. VSCode є популярним інструментом серед розробників, який підтримує багато мов програмування та надає розширення для покращення продуктивності.

- Git - це система керування версіями, яка дозволяє зберігати, відстежувати та спільно працювати з кодом. Git є потужним інструментом для контролю версій та спільної роботи над проектом.

Ці програмні засоби допомагають розробникам зручно працювати над проектом, забезпечуючи зручність у редагуванні коду, керуванні версіями та спільній роботі з командою.

2.6.3. Виклик та завантаження програми

Для виклику та завантаження програми на Angular та Json Server на Windows 10 потрібно виконати наступні кроки:

1. Встановити Node.js:
   - Завантажити та встановити останню версію Node.js з офіційного веб-сайту (https://nodejs.org).
   - Запустити інсталятор та слідувати інструкціям для завершення процесу встановлення Node.js.
2. Встановити Angular CLI [17]:
   - Відкрити командний рядок (Command Prompt) або термінал.
   - Ввести наступну команду для встановлення Angular CLI: npm install -g @angular/cli.
   - Почекати, поки процес встановлення Angular CLI завершиться.
3. Створити новий проект Angular:

- Відкрити командний рядок (Command Prompt) або термінал.
- Перейти до каталогу, де хочете створити новий проект Angular.
- Ввести наступну команду для створення нового проекту: ng new my-app (замінити "my-app" на назву проекту).
- Почекати, поки Angular CLI завершить процес створення проекту.

4. Запуск Angular-додатку:

- Перейти до каталогу вашого нового проекту, ввести команду cd my-app (замінити "my-app" на назву проекту).
- Ввести команду ng serve для запуску веб-сервера розробки Angular.
- Відкрити браузер та перейти за адресою http://localhost:4200, де буде доступний Angular-додаток.

5. Встановити та запустити Json Server:

- Відкрити новий командний рядок (Command Prompt) або термінал.
- Перейти до каталогу проекту Angular, ввести команду cd my-app (замінити "my-app" на назву проекту).
- Встановити Json Server, ввести команду npm install -g json-server.
- Створити JSON-файл за зразком даних (наприклад, db.json) у кореневій папці проекту.
- Запустити Json Server, ввести команду json-server --watch db.json.

Тепер Angular додаток та Json Server повинні бути успішно запущені на системі Windows 10. Angular додаток буде доступний за адресою http://localhost:4200, а Json Server буде доступний за адресою http://localhost:3000.

2.6.4. Опис інтерфейсу користувача

Після запуску веб-застосунку можливо відкрити цей застосунок за адресою http://localhost:4200 в будь-якому веб-переглядачі (Google Chrome, Mozilla Firefox, тощо). При переході за даним посиланням відкриється стартова сторінка «Login» (рис. 2.2.).

Одразу під назвою «BooksAI» розташовується меню з трьома пунктами (основними сторінками веб-додатку): «Login», «Bookshelf», «Search». Перша сторінка «Login» дає можливість користовачу отримати доступ до додатку шляхом введення конкретного імені та пароля. Доступ дається через базу даних напряму у форматі JSON (рис. 2.3.).

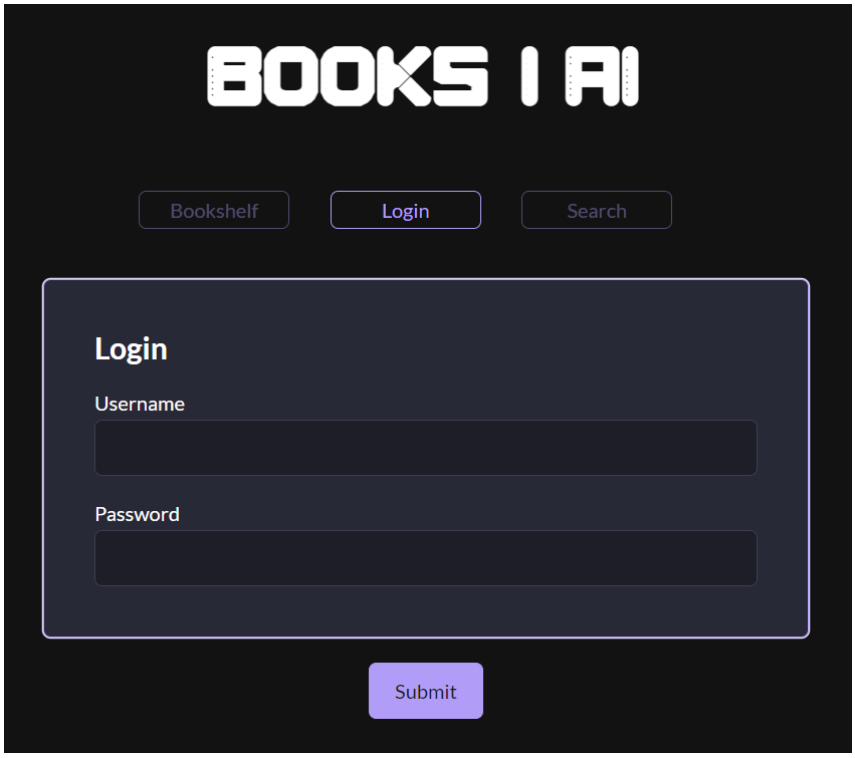

Рис. 2.2. Стартова сторінка додатку

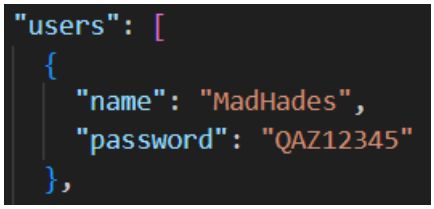

Рис. 2.3. Користувач у базі даних

Після заповнення стартової форми користувач отримує доступ до всіх функцій додатку та автоматично переходить на сторінку «Search» (рис. 2.4.).

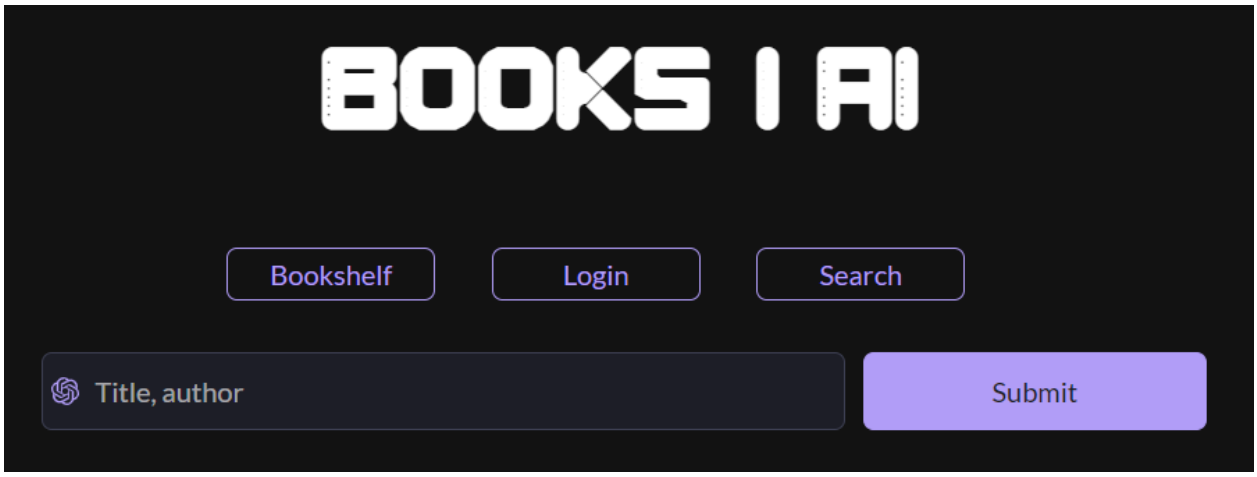

Рис. 2.4. Сторінка для пошуку книг

На цій сторінці користувач може шукати книжку за заданим шаблоном, а саме «Назва книжки, автор» (рис. 2.5.). Пошук можна здійснювати на будь-якій зручній мові (рис. 2.6.).

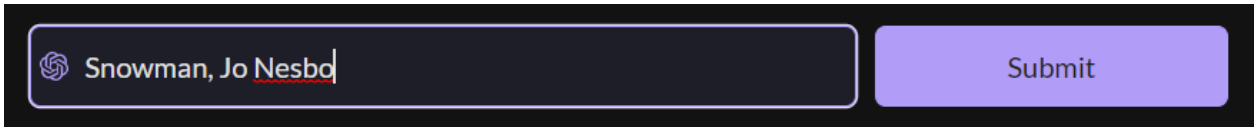

Рис. 2.5. Приклад використання пошуку на англійській мові

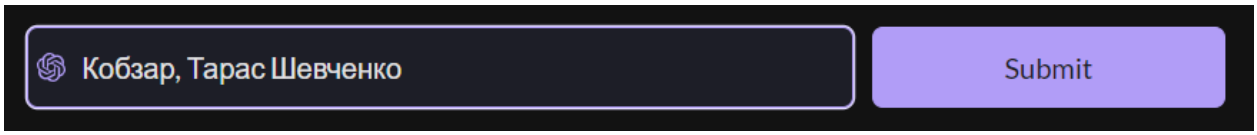

Рис. 2.6. Приклад використання пошуку на українській мові

Коли користувач натискає кнопку «Submit», то починається запит до сервісу OpenAI (рис. 2.7.).

Рис. 2.7. Повідомлення, що запит розпочався

Запит завершується успішно та користувач бачить перед собою 3 книги, які мають максимальне співпадіння з пошуковою стрічкою (рис. 2.8.). На картці з книгою користувач може побачити назву книги, її автора, стислий опис книги та відповідність пошуковій стрічці. Книга з найбільшим співпадінням виділена фіолетовою рамкою (рис. 2.9.). Також на кожній картці є кнопка «Add to list» (рис. 2.10.), яка дозволяє користувачеві додати обрану книгу до своєї книжкової полиці.

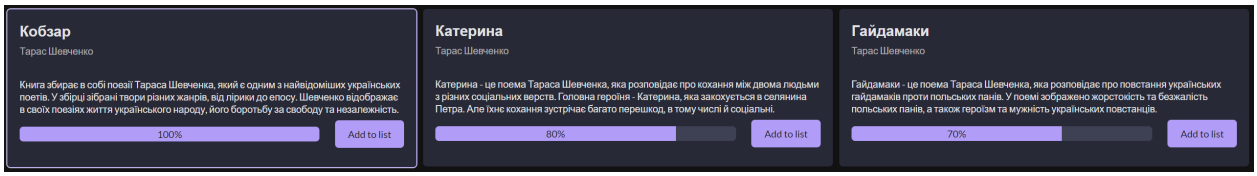

Рис. 2.8. Результат успішного запиту

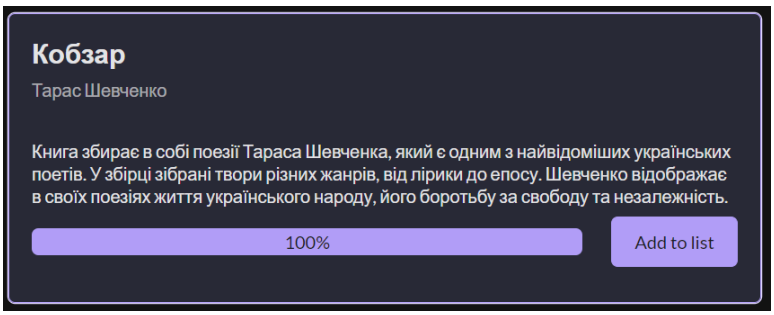

Рис. 2.9. Карточка, яка має найбільше співпадіння

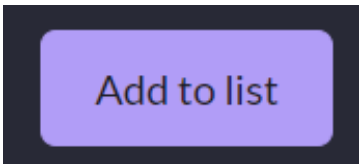

Рис. 2.10. Кнопка, яка додає книгу до полиці

Після того, як користувач натискає дану кнопку, застосунок переміщує користувача на сторінку «Bookshelf» (рис. 2.11.).

На сторінці «Bookshelf» користувач може створити нову полицю (рис. 2.12.) або додати обрану на попередньому етапі книгу до вже існуючої полиці (рис 2.13.).

Слідуючи першому сценарію, тобто, коли користувач натискає кнопку «Create new list», система відкриває на лівій половині екрану форму для створення нової полиці (рис. 2.14.).

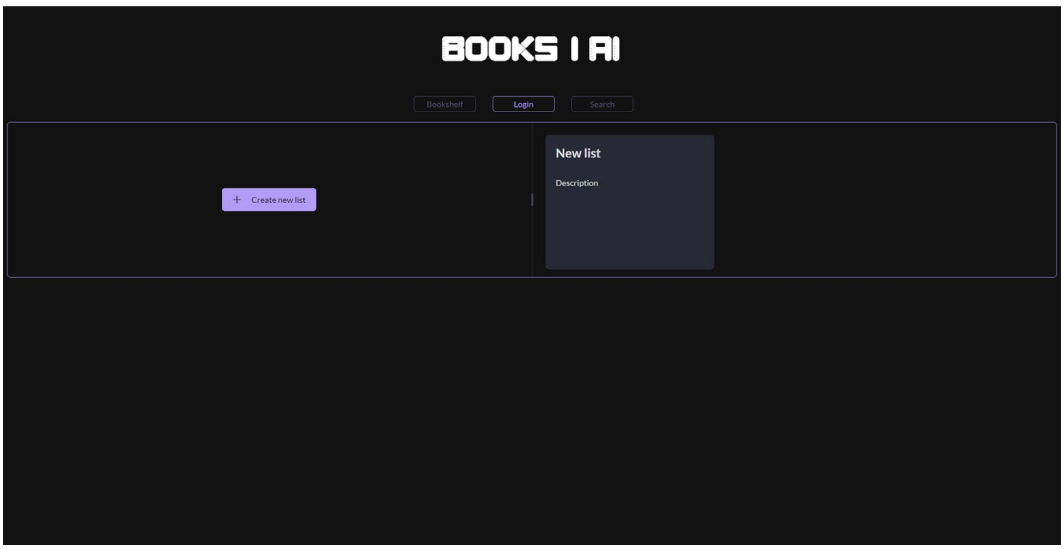

Рис. 2.11. Сторінка для управління полицями

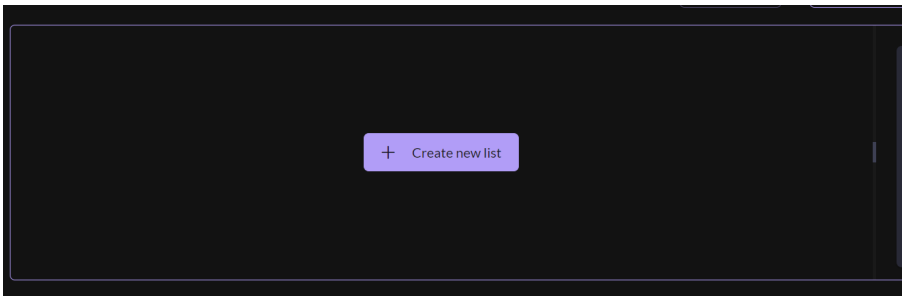

Рис. 2.12. Кнопка для створення нової полиці

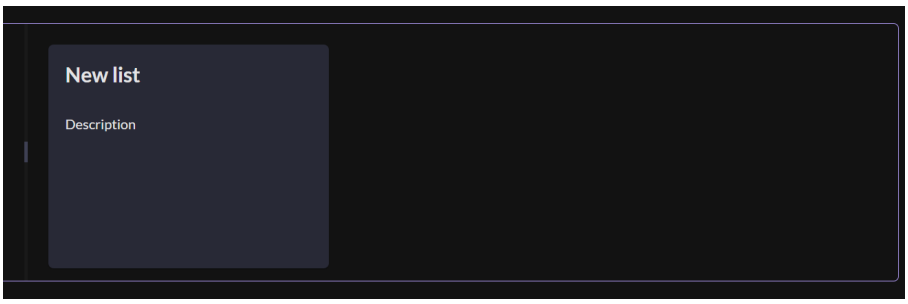

Рис. 2.13. Існуюча полиця користувача

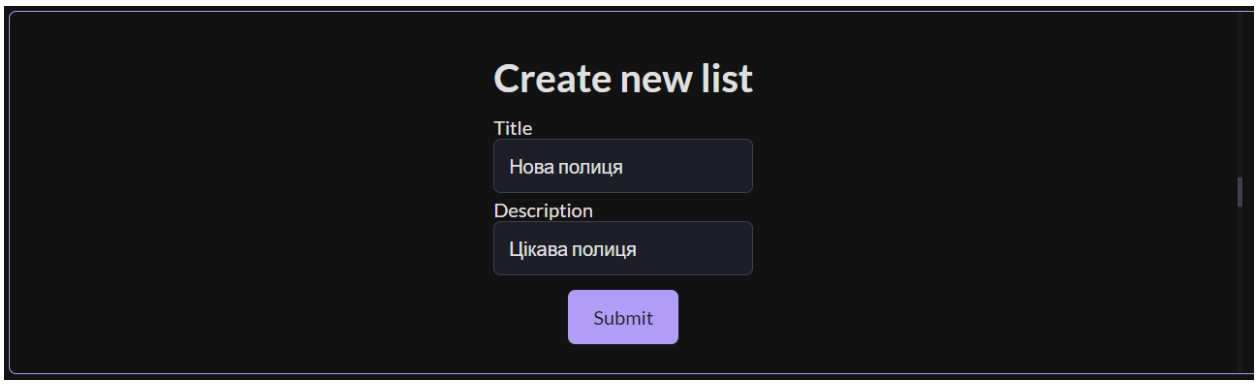

Рис. 2.14. Форма для створення полиці

Користувач заповнює два поля «Назва» та «Опис» полиці та натискає кнопку «Submit» (рис. 2.15.).

Рис. 2.15. Кнопка для створення полиці

Користувач натискає кнопку «Submit», та у правій частині екрану з’являється нова книжкова полиця (рис. 2.16.).

Слідуючи другому сценарію, тобто, коли користувач натискає на полицю у яку хоче додати обрану книу, система автоматично переводить користувача на сторінку з вмістом книжкової полиці (рис. 2.17.).

На актуальній сторінці застосунку користувач может підібрати книги схожі на ті, що вже є в цій полиці. Для цього користувач натискає кнопку «Get book advices» (рис. 2.18.).

Користувач натискає кнопку «Get book advices» та через деякий час (20-40 секунд) отримує список книг подібних до книг у полиці (рис. 2.19.).

Рис. 2.16. Права частина з існуючими полицями

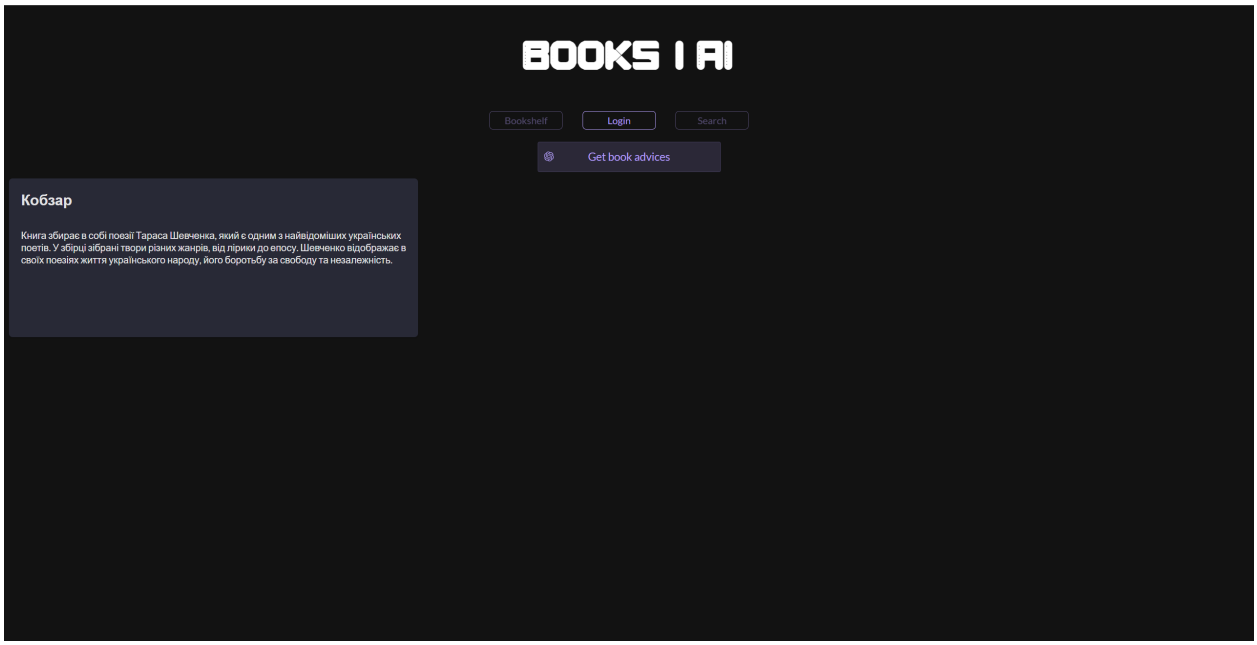

Рис. 2.17. Сторінка з полицею, у яку додано обрану книгу

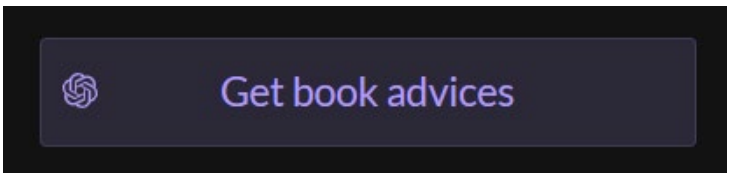

Рис. 2.18. Кнопка для підбору рекомендацій

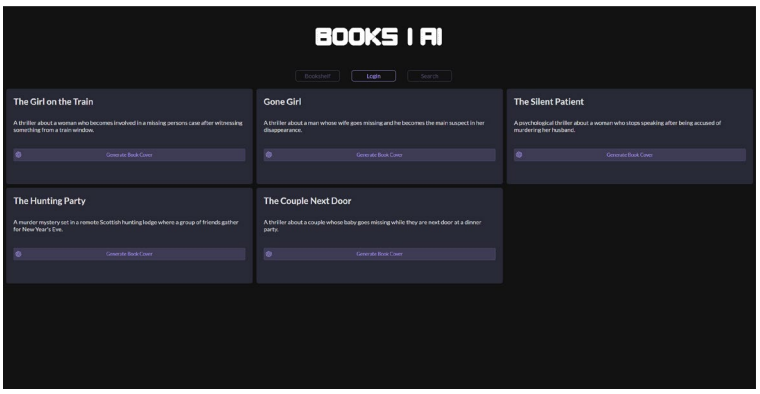

Рис. 2.19. Підібрані рекомендації за полицею з книгою «Snowman, Jo Nesbo»

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки інформаційної системи

Початкові дані:

1. передбачуване число операторів програми –1850;
2. коефіцієнт кореляції програми в ході її розробки – 0,15;
3. коефіцієнт складності програми – 1,6;
4. годинна заробітна плата програміста– 295 грн/год;

Зарплата програміста (Middle Frontend developer) була обчислена на основі даних «Української спільноти програмістів (DOU)» .

Наприкінці 2022 року, інформація за грудень, зарплата Middle FE розробника починається від 1200$ до 2500$. За даними тієї ж спільноти, медіанна заробітна плата по Україні становить 1400$ у місяць.

Враховуючі курс валют НБУ на початок січня 2023 року один долар США дорівнює 36,9 грн, тому середня зарплата в гривнях дорівнює 52000 грн. При графіку 40 робочих годин на тиждень (176 годин/місяць) зарплата за годину буде становити близько 295 грн.

5. коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,1;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;
7. вартість машино-години ЕОМ – 12 грн/год.

Оскільки для цього проекту потрібна велика потужність ПК для розгортання на локальному сервері, гарним рішенням буде оренда комп’ютера на час розробки додатку. Вартість ноутбуку, що підходить для виконання поставленої задачі на місяць починається від 10000 грн. Оренда такого пристрою складає 628 грн. Оренда не передбачає повернення техніки на

вихідних або у неробочій час власнику, але робота на ноутбуці проходить тільки у робочій час, тож вартість оренди погодинно вираховується на основі 176 робочих годин на місяць. Тож вартість ЕОМ за годину роботи буде становити 12 грн. В цю вартість входить ремонт за гарантією.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин} \quad (3.1)$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

t_n-витрати праці на програмування по готовій блок-схемі;

$t_{отл}$-витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p) \quad (3.2)$$

де q - передбачуване число операторів (1850);

C - коефіцієнт складності програми (1,6);

p - коефіцієнт кореляції програми в ході її розробки (0,15).

Звідси умовне число операторів в програмі:

$$Q = 1{,}8 \cdot 628 \cdot (1 + 0{,}2) = 1356{,}48$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин} \qquad (3.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 4 до 8 років він складає 1,4.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,3). З урахуванням коефіцієнта кваліфікації k = 1,4, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (1356{,}48 \cdot 1{,}3) / (75 \cdot 1{,}4) = 16{,}79 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k} \text{ людино-годин} \qquad (3.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), отримаємо:

$$t_a = 1356{,}48 / (20 \cdot 1{,}4) = 48{,}45 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25) \cdot k}, \text{ людино-годин} \quad (3.5)$$

$$t_n = 1356{,}48 / (25 \cdot 1{,}4) = 38{,}75 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k}, \text{ людино-годин} \quad (3.6)$$

$$t_{отл} = 1356{,}48 / (5 \cdot 1{,}4) = 193{,}78 \text{ людино-годин.}$$

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}, \text{ людино-годин} \quad (3.7)$$

$$t^k_{отл} = 1{,}5 \cdot 193{,}78 = 290{,}67 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

(3.8)

$$t_{д} = t_{др} + t_{до}$$, людино-годин

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k}$$, людино-годин (3.9)

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 \cdot t_{др}$$, людино-годин (3.10)

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 1356,48 / (20 · 1,4) = 48,45 людино-годин.

$t_{до}$ = 0,75 · 48,45 = 36,34 людино-годин.

$t_{д}$ = 48,45 + 36,34 = 84,79 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 16,79 + 48,45 + 38,75 + 290,67 + 84,79 = 529,45 людино-годин.

3.2. Розрахунок витрат на створення сайту

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн} \quad (3.12)$$

де t - загальна трудомісткість, людино-годин,

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 125 грн / год, отримуємо:

$$З_{ЗП} = 290,67 \cdot 140 = 40693,8 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (12 грн/год).

Підставивши в формулу (3.14) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 290,67 \cdot 12 = 3488,04 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$K_{ПО}$ = 40693,8+ 3488,04 = 44181,81 грн.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.} \quad (3.14)$$

де B_k- число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси, за формулою (3.14) витрати на створення програмного продукту:

T = 529,45 / (1·176) ≈ 3 міс.

Висновок: веб-застосунок було розроблено з метою здійснення забезпечення користувачам доступу до книжок та рекомендацій по книжковим полицям. На розробку веб-застосунку піде близько 529,45 годин, або 3 місяці. Вартість даного сайту становитиме близько 44181,81 грн.. Цей термін пов'язаний зі значним числом строків коду CSS/SCSS, і включає час на написання документації-опису, необхідної для створення додатку, дослідження і розробку алгоритму вирішення поставленого завдання.

ВИСНОВКИ

В результаті написання кваліфікаційної роботи було створено веб-застосунок для любителів книг. В ході виконання кваліфікаційної роботи були виконані такі завдання:

- проведення детального аналізу вимог до інформаційної системи, зокрема до книжного сервісу;
- проектування бази даних;
- розробка користувацького інтерфейсу;
- реалізація функціональності;
- тестування;
- інтеграція з одним із провідних сервісів штучного інтелекту.

Розроблена система має прикладний характер та дозволяє користувачам отримувати рекомендації за книгами, які сподобалися.

Подальші пункти для розвитку застосунку: введення можливості для створення нового користувача через інтерфейс застосунку, введення можливості видалення книжкової полиці та книжки з полиці тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Посібник використання HTML: https://w3schoolsua.github.io/html/index.html#gsc.tab=0
2. Посібник використання SCSS: https://sass-lang.com/documentation/syntax#scss
3. Посібник використання Javascript: https://developer.mozilla.org/ru/docs/Web/JavaScript
4. Посібник по бібліотеці PrimeNG: https://primeng.org/
5. Посібник по JSON Server: https://www.npmjs.com/package/json-server
6. Інструкція з використнння JSON: https://www.w3schools.com/whatis/whatis_json.asp
7. Посібник використання TypeScript: https://www.typescriptlang.org/
8. Посібник використання RxJS: https://rxjs.dev/
9. Посібник використання Angular: https://angular.io/
10. Посібник використання Bootstrap: https://getbootstrap.com/
11. Посібник використання OpenAI API: https://platform.openai.com/docs/api-reference
12. Посібник по архітектурам застосунків: https://openclassrooms.com/en/courses/6397806-design-your-software-architecture-using-industry-standard-patterns/6896176-layered-architecture
13. Керівництво по webpack: https://webpack.js.org/
14. Керівництво по життевому циклу сторінок веб-застосунку: https://developer.chrome.com/blog/page-lifecycle-api/#overview-of-page-lifecycle-states-and-events
15. Керівництво по eventloop: https://blog.xnim.me/event-loop-and-render-queue
16. Керівництво по RESTful API: https://www.techtarget.com/searchapparchitecture/definition/RESTful-

API#:~:text=A%20RESTful%20API%20is%20an,deleting%20of%20operations%20concerning%20resources.
17. Керівництво по Angular CLI: https://www.educba.com/angular-cli/
18. Стаття з порівнянням архітектурних підходів для побудови сучасного веб-застосунку: https://www.angularminds.com/blog/article/mvc-vs-mvp-mvvm.html
19. Стаття з інформацією про протокол обміну даними HTTPS: https://www.cloudflare.com/learning/ssl/what-is-https/
20. Стаття про SPA (single-page application): https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application?spz=learn_var

ДОДАТОК А

КОД ПРОГРАМИ

app.component.html

```
<h1>Books | AI</h1>
<books-navigation></books-navigation>
<router-outlet></router-outlet>
<p-toast></p-toast>
```

app.component.scss

```
@import url('https://fonts.cdnfonts.com/css/pluto-0');

h1 {
  text-align: center;
  font-size: 80px;
  font-family: 'Pluto 0', sans-serif;
}

::ng-deep p-toast {
  .p-toast {
    opacity: 1 !important;
    max-width: 25rem;
    width: auto;

    &-message-custom {
      font-size: 18px;
      background-color: var(--highlight-bg);
      color: var(--highlight-text-color);
    }
```

```
    &-top-right {
      right: 10px;
      top: 10px;
    }
  }
}

.logo {
  margin: 80px 0;
  font-size: 80px;
  font-family: 'Pluto 0', sans-serif;
  text-align: center;
}
```

app.component.ts

```
import { Component } from '@angular/core';
import { MenuItem, MessageService } from 'primeng/api';
import { AdvicesService } from './services/advices.service';
import { filter, fromEvent, map, tap } from 'rxjs';
import { LoginService } from './services/login.service';

@Component({
  selector: 'books-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss'],
  providers: [MessageService]
})
export class AppComponent {
```

```
  constructor(private messageService: MessageService,
private advicesService: AdvicesService, private
loginService: LoginService) { }


  ngOnInit(): void {
    this.checkRequestStatus();
  }


  ngAfterViewInit(): void {
    this.loginService.userFromStorage$.subscribe(data => {
      if (data) {
        this.loginService.isUserLogged$.next(true);
      }
    });
  }


  checkRequestStatus(): void {
    this.advicesService.requestStarts$.subscribe(data => {
      if (data) {
        this.messageService.add({
          severity: 'info',
          detail: 'Request is processing...',
          sticky: true
        })
      } else {
        this.messageService.clear();
        this.messageService.add({
          severity: 'success',
          detail: 'All ready!',
```

```
          life: 3000
        })
      }
    })
  }
}
```

book-lists.component.html

```
<article>
  <div class="book__search_wrapper">
    <input class="book__search" type="text" pInputText
placeholder="Title, author" [(ngModel)]="searchQuery"/>
    <p-button class="book__search_submit" label="Submit"
(onClick)="onBookFind()"></p-button>
  </div>
  <div class="book__list">
    <p-card
      *ngFor="let book of (advicesService.findedBooks$ |
async) || [] | sort:'desc':'equalityToRequestInPercents';
let i = index"
      class="book__card"
      [ngClass]="{'book__card_first': i === 0}"
      [header]="book.title"
      [subheader]="book.author"
    >
      <p class="m-0">{{
book.shortDescriptionWithoutForbiddenWords}}</p>
      <div class="flex">
        <p-progressBar class="book__equality"
pTooltip="Match with the search string"
tooltipPosition="bottom"
[value]="book.equalityToRequestInPercents"></p-progressBar>
```

```
        <p-button class="book__add" label="Add to list"
(onClick)="onAddToList(book)"></p-button>
      </div>
    </p-card>
  </div>
</article>
```

book-lists.component.scss

```
@import url('https://fonts.cdnfonts.com/css/pluto-0');

article {
  height: 100%;
  display: flex;
  flex-direction: column;
  align-items: center;
}

h1 {
  font-size: 80px;
  font-family: 'Pluto 0', sans-serif;
}

.book {
  &__list {
    display: grid;
    margin-top: 10px;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: 0.15fr;
    grid-gap: 10px;
  }
```

```
  &__cover {
    display: flex;
    width: 100px;
  }

  &__add {
    width: 18%;
  }

  &__search {
    width: 70%;
    padding-left: 30px;
    background-image:
url('../../../assets/images/logoGPT.svg');
    background-position: 5px center;
    background-size: 16px;
    background-repeat: no-repeat;

    &_wrapper {
      width: 33%;
      display: flex;
      margin: 10px 0;
    }

    &_submit {
      width: 30%;
      margin-left: 10px;
    }
```

```
  }

  &__equality {
    width: 78%;
  }
}

::ng-deep {
  .p-button {
    width: 100%;
  }

  .book__card {
    height: 260px;

    &_first {
      .p-card-body {
        border: 1px solid #b19df7;
        box-shadow: 0 0 0 1px #e0d8fc;
        border-radius: 6px;
      }
    }
  }
}

.flex {
  height: 30px;
  display: flex;
  align-items: center;
```

```
  justify-content: space-between;

  &_column {
    display: flex;
    flex-direction: column;
    justify-content: center;
    width: 78%;

    p {
      text-align: right;
      margin: 0;
    }
  }
}
```

book-lists.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { ApiService } from 'src/app/api/api.service';
import { IList, IListEntry, ISeed } from
'src/app/models/api.model';
import { AdvicesService } from
'src/app/services/advices.service';
import { MessageService } from 'primeng/api';
import { BookListService } from 'src/app/services/book-
list.service';
import { IBookFromAI } from 'src/app/models/book-ai.model';

@Component({
  selector: 'books-lists',
  templateUrl: './book-lists.component.html',
```

```
  styleUrls: ['./book-lists.component.scss'],
  providers: [MessageService]
})
export class BookListsComponent {
  title = 'books';
  baseUrl = 'http://openlibrary.org';
  userName = 'pavel_borodin';
  books: IListEntry[] = [];
  lists: any[] = [];
  isBooksVisible = false;
  searchQuery: string = '';

  constructor(
    public api: ApiService,
    private router: Router,
    public advicesService: AdvicesService,
    private messageService: MessageService,
    private listsService: BookListService
  ) { }

  ngOnInit(): void {
    this.api.getUserLists().subscribe((data: IList) => {

this.api.readList(data.entries[0].url).subscribe((data: any)
=> {
        this.lists.push(data);
      })

      this.api.getSeedsByList(data).subscribe((seed: ISeed)
=> {
```

```
        this.books = seed.entries;
      })
    })
  }

  onListClick(name: string): void {
    this.isBooksVisible = !this.isBooksVisible;
  }

  navigateOnAdvicesPage(event: MouseEvent): void {
    event.stopPropagation();
    let booksForSearch = '';
    let booksForSearchArr: string[] = [];

    this.books.forEach(book => {
      booksForSearchArr.push(book.title);
    });

    booksForSearch = booksForSearchArr.join(', ');

    this.advicesService.getBooksAdvices(booksForSearch);

    this.advicesService.reloaderForAIRequest$.subscribe(()
=> {
      this.advicesService.getBooksAdvices(booksForSearch);
    })

    this.advicesService.similarBooks$.subscribe((data) => {
      this.advicesService.similarBooks = JSON.parse(data);
```

```
      this.router.navigateByUrl('/advices');
    })
  }

  onBookFind(): void {
    this.advicesService.findBooks(this.searchQuery);
  }

  onAddToList(book: IBookFromAI): void {
    this.listsService.isAddToListBtnClicked = true;
    this.listsService.selectedBook = book;
    this.router.navigateByUrl('/lists');
  }
}
```

bookshells.component.html

```
<p-splitter [style]="{ height: '300px' }" styleClass="mb-5">
  <ng-template pTemplate>
    <div class="book__lists_wrapper
book__lists_wrapper_form">
      <p-button *ngIf="!isFormOpened" class="book__add"
label="Create new list" (onClick)="onCreateNewList()"><i
class="pi pi-plus"></i></p-button>
      <form *ngIf="isFormOpened"
[formGroup]="createListForm">
        <h1>Create new list</h1>
        <div class="book__lists-form__row">
          <label for="title">Title</label>
          <input id="title" [ngClass]="{'invalid': false}"
pInputText type="text" formControlName="title"
autocomplete="off"/>
```

```
          <p *ngIf="false" class="login-form__error">No such
user in data base</p>
        </div>
        <div class="book__lists-form__row">
          <label for="description">Description</label>
          <input id="description" [ngClass]="{'invalid':
false}" pInputText type="text" formControlName="description"
autocomplete="off"/>
          <p *ngIf="false" class="login-form__error">Your
password is not right!</p>
        </div>
      </form>
      <p-button *ngIf="isFormOpened" class="book__lists-
form__submit" label="Submit" (onClick)="onSubmit()"></p-
button>
    </div>
  </ng-template>
  <ng-template pTemplate>
      <div class="book__lists_wrapper">
        <p-card *ngFor="let list of lists"
class="book__card" [header]="list.title"
(click)="onAddToList(list.id || 0)">
          <p class="m-0">{{ list.description }}</p>
        </p-card>
      </div>
  </ng-template>
</p-splitter>
```

bookshells.component.scss

```
.book {
  &__add {
    margin: 12px;
```

```
  .pi-plus {
    padding-right: 1.25rem;
  }
}

&__advices {
  display: grid;
  margin-top: 10px;
  grid-template-columns: 1fr 1fr 1fr;
  grid-gap: 10px;
}

&__cover {
  margin-top: auto;
}

&__lists {
  &_wrapper {
    display: flex;
    justify-content: space-between;
    width: 100%;
    margin: 12px;

    &_form {
      display: flex;
      flex-direction: column;
      justify-content: center;
      align-items: center;
```

```
        h1 {
          margin: 12px 0;
        }
      }
    }

    &-form {
      &__submit {
        margin-top: 12px;
      }

      &__row {
        display: flex;
        flex-direction: column;
        justify-content: space-between;
        margin-top: 4px;
      }
    }
  }

  &__card {
    width: 33%;
    margin: 12px;
    cursor: pointer;
  }
}

::ng-deep {
```

```
  .book__card {
    .p-card {
      height: 100% !important;
    }
  }
}
```

bookshells.component.ts

```
import { ChangeDetectorRef, Component } from
'@angular/core';
import { FormBuilder, FormGroup, Validators } from
'@angular/forms';
import { Router, RoutesRecognized } from '@angular/router';
import { BehaviorSubject, filter, pairwise } from 'rxjs';
import { ApiService } from 'src/app/api/api.service';
import { IBookFromAI } from 'src/app/models/book-ai.model';
import { IList } from 'src/app/models/list.model';
import { AdvicesService } from
'src/app/services/advices.service';
import { BookListService } from 'src/app/services/book-
list.service';


@Component({
  selector: 'books-bookshells',
  templateUrl: './bookshells.component.html',
  styleUrls: ['./bookshells.component.scss']
})
export class BookshellsComponent {
  lists: IList[] = [];
  isFormOpened = false;
  createListForm: FormGroup = {} as any;
```

```
  canAddToList = false;

  constructor(
    public advicesService: AdvicesService,
    private api: ApiService,
    private listsService: BookListService,
    private fb: FormBuilder,
    private router: Router,
    private cdr: ChangeDetectorRef
  ) { }

  ngOnInit(): void {
    this.advicesService.similarBooks$.subscribe(data => {
      console.log(JSON.parse(data));
    })

    this.getUserLists();
  }

  ngAfterViewInit(): void {
    this.router.events
    .pipe(filter((evt: any) => evt instanceof
RoutesRecognized), pairwise())
    .subscribe((events: any) => {
      if (this.listsService.isAddToListBtnClicked &&
events[0].urlAfterRedirects === '/search') {
        this.canAddToList = true;
      }
    });
```

```
  }


  onCoverGenerateClick(event: MouseEvent, book:
IBookFromAI): void {
    event.stopPropagation();


    this.advicesService.generateBookCover(`Create book cover
for '${book.title}', author ${book.author}. Book description
-
${book.shortDescriptionWithoutForbiddenWords}`).subscribe(da
ta => {
      // saveAs(data.data.data[0].url, 'cover.jpg')
      console.log(data.data.data[0].url);
    });
  }


  onCreateNewList(): void {
    this.isFormOpened = true;
    this.onFormCreate();
  }


  getUserLists(): void {
    const user = JSON.parse(sessionStorage.getItem('user')
as any);

this.listsService.getUserLists(user.name).subscribe((data:
any) => {
      console.log(data);
      this.lists = data;
    })
  }
```

```
  onFormCreate(): void {
    this.createListForm = this.fb.group({
      title: ['', Validators.required],
      description: ['']
    })
  }

  onSubmit(): void {
    this.listsService.postUserList({
      title: this.createListForm.value.title,
      description: this.createListForm.value.description,
      userName: JSON.parse(sessionStorage.getItem('user') as
any).name,
      books: []
    }).subscribe(() => {
      this.getUserLists();
    });
  }

  onAddToList(listId: number): void {
    console.log("Ok");

this.listsService.postBookToList({...this.listsService.selec
tedBook, listId: listId}).subscribe(() => {
      this.listsService.currentBookListId = listId;
      this.router.navigateByUrl('/list');
    });
  }
}
```

advices.component.html

```
<article class="book__advices">
  <p-card *ngFor="let book of advicesService.similarBooks" class="book__card" [header]="book.title">
    <p class="m-0">{{ book.shortDescriptionWithoutForbiddenWords}}</p>
    <button class="btn btn_gpt book__cover" (click)="onCoverGenerateClick($event, book)">Generate Book Cover</button>
  </p-card>
</article>
```

advices.component.scss

```
.book {
  &__advices {
    display: grid;
    margin-top: 10px;
    grid-template-columns: 1fr 1fr 1fr;
    grid-gap: 10px;
  }

  &__cover {
    margin-top: auto;
  }
}
```

advices.service.ts

```
import { Injectable } from '@angular/core';
import { Observable, Subject, from } from 'rxjs';
import { isJsonString } from 'src/helpers/helpers';
import { OpenAIApi } from 'openai/dist/api';
```

```
import { Configuration } from 'openai';
import { IBookFromAI } from '../models/book-ai.model';
import { ApiService } from '../api/api.service';
import { apiKey } from 'src/helpers/api-key';

@Injectable({
  providedIn: 'root'
})
export class AdvicesService {
  similarBooks$ = new Subject<string>();
  reloaderForAIRequest$ = new Subject();
  similarBooks: IBookFromAI[] = [];
  findedBooks$ = new Subject<any[]>();
  requestStarts$ = new Subject<boolean>();

  configuration = new Configuration({
    organization: "org-TyyH03CDnaoMqj1UwUmi7Dha",
    apiKey: apiKey,
  });
  openai: OpenAIApi = new OpenAIApi(this.configuration);

  constructor(private api: ApiService) { }

  getBooksAdvices(books: string): void {
    const prompt = `Create json array of objects in format
{title, author, shortDescriptionWithoutForbiddenWords} books
similar to books like this ${books}? Only array as text.
Dont return the same books!`

    const completions = this.openai.createChatCompletion({
```

```
      model: 'gpt-3.5-turbo',
      messages: [{role: 'user', content: prompt}],
      temperature: 1
    });

    this.requestStarts$.next(true);

    from(completions).subscribe((data: any) => {
      this.requestStarts$.next(false);
      const response = data.data.choices[0].message?.content
as string;
      isJsonString(response) ?
this.similarBooks$.next(response) :
this.reloaderForAIRequest$.next(null);
    })
  }

  generateBookCover(prompt: string): Observable<any> {
    const generation = this.openai.createImage({
      prompt: prompt,
      n: 1,
      size: "256x256"
    });

    return from(generation);
  }

  findBooks(userInput: string): void {
    const prompt = `Create json array of objects in format
{title, author, shortDescriptionWithoutForbiddenWords(min:
```

```
200 symbols, max: 250 symbols), equalityToRequestInPercents}
books by this {name, author}: ${userInput}? Only array as
text that can be converted to JSON format!!! No more text,
only array as text!!! Provide 3 books in array`;


    const completions = this.openai.createChatCompletion({
      model: 'gpt-3.5-turbo',
      messages: [{role: 'user', content: prompt}],
      temperature: 0.1
    });


    this.requestStarts$.next(true);


    from(completions).subscribe((data: any) => {
      this.requestStarts$.next(false);
      const response = data.data.choices[0].message?.content
as string;
      const regex = /(.*):(.*)/;
      const books = response.match(regex) || '';
      isJsonString(response) ?
this.findedBooks$.next(JSON.parse(response)) :
this.findedBooks$.next(JSON.parse(books[0]));
    })
  }
}
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра на тему:
Розробка клієнтської частини веб-застосунку для книжкового сервісу засобами Angular
студента групи 122-19-3
Бородіна Павла Євгеновича

Керівник економічного розділу
Зав. каф. ПЕП та ПУ, д.екон.н. Вагонова О.Г.

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я	**Опис**
Пояснювальні документи	
Бородін Павло.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Бородін Павло.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
booksai.zip	Архів. Містить коди програми.
Презентація	
Презентація БородінПЄ.pptx	Презентація кваліфікаційної роботи.