

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка
до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ВЕБСАЙТ ПІДВИЩЕНОЇ НАДІЙНОСТІ ІЗ
ІНТЕЛЕКТУАЛЬНИМ ЧАТ-БОТОМ

Виконав: студент 2 курсу, групи КНТ-622м
спеціальності _____

123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Спеціалізовані комп'ютерні системи

БІЛАШ Д.І.

(ПРИЗВИЩЕ та ініціали)

Керівник КУДЕРМЕТОВ Р.К.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЗІНА Г.Л.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) «Спеціалізовані комп'ютерні системи»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“14” листопада 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

БІЛАША Данііл Ігоровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Вебсайт підвищеної надійності із інтелектуальним чат-ботом

керівник проєкту (роботи) к. т. н., доцент, КУДЕРМЕТОВ Равіль Камілович

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) програмні засоби створення вебсайтів

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

1) Аналіз предметної області;

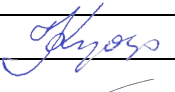
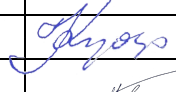
2) Проєктування системи;

3) Реалізація вебсайту;

4) Результати роботи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проекту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	КУДЕРМЕТОВ Р.К., зав.каф.		
Нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	02.10.2023р.	
2	Розробка структурної схеми системи	10.10.2023р.	
3	Розробка методології дослідження та розробки	16.10.2023р.	
4	Вибір технологій для розробки вебсайту	25.10.2023р.	
5	Розробка вебсайту з урахуванням вимог до надійності та безпеки	05.11.2023р.	
6	Розробка інтелектуального чат-бота та його інтеграція з вебсайтом	15.11.2023р.	
7	Проведення тестування вебсайту та чат-бота, збір даних та аналіз результатів	28.11.2023р.	
8	Оформлення отриманих результатів в ПЗ	06.12.2023р.	

Студент


(підпис)

Даніїл БІЛАШ
(ім'я, ПРИЗВИЩЕ)

Керівник проекту (роботи)


(підпис)

Равіль КУДЕРМЕТОВ
(ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 70 с., 25 рис. , 17 джерел.

ВЕБСАЙТ, КОРИСТУВАЧ, РЕЄСТРАЦІЯ, АВТОРИЗАЦІЯ, ВЕБРОЗРОБКА, ВЕБДИЗАЙН

Об'єкт дослідження – якісні вебсайти з різнобічним функціоналом, у тому числі із застосування технологій штучного інтелекту.

Предмет дослідження – засоби покращення якості, надійності та функціональності вебсайту.

Мета дослідження - покращити якість та надійність вебсайту шляхом вдосконалення та оптимізації його функцій.

У першому розділі розглянуто особливості предметної області, досліджено методи підвищення захисту вебсайтів та розглянуто шляхи інтеграції чат-ботів до вебсайтів.

У другому розділі проведено проектування системи, вибір основних технологій для розробки. Побудовано схеми сторінок для вебсайту.

У третьому розділі наведено реалізацію спроектованої системи з використання технологій Node.js, Mongoose, Express.js, Handlebars.js, bcryptjs.

У четвертому розділі наведено результати роботи вебсайта.

У результаті виконаної роботи було успішно інтегровано інтелектуального чат-бота на вебсайті, що призвело до значного покращення взаємодії з користувачами. Додавання чат-бота дозволило швидко та ефективно відповідати на запитання користувачів, надавати необхідну інформацію та підтримувати взаємодію в режимі реального часу.

Отримані результати свідчать про поліпшення надійності та зручності вебсайту, що, в свою чергу, позитивно впливає на загальне враження користувачів та їхню задоволеність обслуговуванням. Інтелектуальний чат-бот сприяє збільшенню ефективності взаємодії, забезпечуючи високий рівень обслуговування

ABSTRACT

Explanatory note to the master's work: 70 p., 25 figures, 17 sources.

WEBSITE, USER, REGISTRATION, AUTHORIZATION, WEB DEVELOPMENT, WEB DESIGN

Research Object – Qualitative websites with diverse functionality, including the implementation of artificial intelligence technologies.

Research Subject – Means of improving the quality, reliability, and functionality of a website.

Research Goal – Enhance the quality and reliability of the website by refining and optimizing its functions.

In the first section, the fundamental aspects of development for increased reliability and the incorporation of artificial intelligence into our project are discussed. The analysis covers the key design aspects of our application.

The second section involves the design of the system, the selection of core technologies for development, and the construction of page schematics for the website.

The third section presents the implementation of the designed system using Node.js, Mongoose, Express.js, Handlebars.js, and bcryptjs technologies.

In the fourth section, the results of the website's operation are outlined.

As a result of the completed work, the integration of an intelligent chatbot on the website was successfully accomplished, leading to a significant improvement in user interaction. The addition of the chatbot enabled prompt and effective responses to user queries, providing necessary information and facilitating real-time interaction.

The obtained results indicate an enhancement in the website's reliability and user convenience, positively impacting overall user experience and satisfaction with the service. The intelligent chatbot contributes to increased interaction efficiency, ensuring a high level of service.

ЗМІСТ

Скорочення та умовні позначки	8
Вступ	9
1 Аналіз предметної області	11
1.1 Огляд предметної області	11
1.2 Дослідження методів для підвищення захисту вебсайту	13
1.3 Інтеграція чат-ботів в свій вебсайт	16
2 Проєктування системи	21
2.1 Вимоги для функціоналу системи	21
2.2 Розробка структурної схеми	22
2.3 Вибір технологій для розробки	23
2.3.1 Мова програмування JavaScript	23
2.3.2 Середовище виконання Node.js	25
2.3.3 Мова розмітки HTML та стилізації CSS	26
2.3.4 Інструменти для покращення вебсайт	29
3 Реалізація проєкту	32
3.1 Реалізація серверної частини	32
3.1.1 Головний файл вебсайту index.js	32
3.1.2 Реалізація маршрутизації	34
3.1.3 Реалізація функцій реєстрації та авторизації	36
3.1.4 Налаштування маршрутів для авторизації	40
3.1.6 Створення маршрутів для вебсайту BookHaven.js	41
3.1.7 Моделі для роботи з базою даних	42

3.2 Реалізація дизайну вебсайту	43
3.2.1 Головна сторінка додатку	43
3.2.2 Форма реєстрації користувачів	48
3.2.3 Відображення детальної інформації про книги	49
3.2.4 Реалізація форми для додавання книг до бази даних.....	50
3.2.5 Форма для запитання у ChatGPT.....	52
3.2.6 Перелік книг, які знаходяться в базі даних	53
3.2.7 Форма авторизації.....	55
3.3 Створення бази даних MongoDB.....	57
4 Результати роботи.....	61
Висновки.....	68
Перелік джерел посилання	69

Скорочення та умовні позначки

AI - Штучний інтелект;

API - Інтерфейс програмування застосунків;

ML - Машинне навчання;

HTML - Мова розмітки гіпертексту;

CSS - Каскадні таблиці стилів;

HTTP - Протокол передачі гіпертексту;

SQL - Мова структурованих запитів;

SEO - Оптимізація для пошукових систем;

DB - База даних.

ВСТУП

У сучасній епісі цифрових технологій вебсайти стають ключовими платформами для взаємодії компаній, організацій та окремих осіб із їхньою аудиторією, надання інформації, товарів та послуг. Забезпечення надійності вебсайту є завданням вищого пріоритету, оскільки будь-які проблеми у продуктивності можуть викликати розчарування користувачів, втрату доходу та негативний вплив на репутацію бренду.

Однією з ключових відповідей на цей виклик стає використання інтелектуальних чат-ботів, базованих на штучному інтелекті (AI). Інтелектуальні чат-боти можуть взаємодіяти з користувачами, надаючи швидку та персоналізовану підтримку в реальному часі, що відповідає росту очікувань користувачів.

Вибір теми магістерської роботи, присвяченої вдосконаленню якості та надійності вебсайту через інтеграцію інтелектуального чат-бота, є актуальним і відповідає сучасним тенденціям в галузі веброзробки та штучного інтелекту.

Таким чином обрана тема магістерської роботи є актуальною.

Мета дослідження - покращити якість та надійність вебсайту шляхом вдосконалення та оптимізації його функцій.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- проаналізувати предметну область та наявні методи для підвищення захисту та інтеграції чат-боту у вебсайт;
- визначити вимогу до розроблюваного проєкту;
- спроектувати вебсайт;
- обрати необхідне програмне забезпечення для створення вебсайту;
- реалізувати вебсайт із інтегрованим інтелектуальним чат-ботом;
- проаналізувати отримані результати.

Проект включає глибокий аналіз предметної області, щоб визначити найбільш ефективні методи підвищення безпеки та інтеграції інтелектуального чат-бота у вебсайт. Результати цього аналізу дозволили визначити оптимальні стратегії для досягнення поставленої мети.

Визначення вимог було кроком, важливим для успішної реалізації проекту. Враховуючи очікування користувачів та вимоги до надійності, було визначено конкретні функціональні та технічні вимоги, які повинен був відповідати вебсайт із чат-ботом.

Процес проектування вебсайту включав в себе створення ергономічного та візуально привабливого інтерфейсу, а також планування ефективного впровадження інтелектуального чат-бота. Обрано необхідне програмне забезпечення, щоб забезпечити оптимальну продуктивність і відповідність вимогам проекту.

Реалізація вебсайту із інтегрованим інтелектуальним чат-ботом була важливим етапом, де використовувалися сучасні методи розробки веб-додатків та програмування чат-ботів. Отримані результати були піддані аналізу, щоб визначити ефективність та вплив на якість та надійність вебсайту.

Проект залишається відкритим для подальших можливостей розвитку, таких як розширення функціоналу чат-бота та оптимізація роботи вебсайту. Отримані знання та досвід можуть служити основою для подальших досліджень у галузі веброзробки та використання штучного інтелекту для покращення взаємодії з користувачами в онлайн-середовищі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд предметної області

У наш час важко переоцінити роль вебсайту для різних сфер життя, оскільки він відкриває доступ до інформації з усього світу. Деякі підприємства можуть ігнорувати онлайн-присутність, але більшість людей не погодяться з цим. Створення хоча б невеликого вебсайту є важливим кроком для активного розвитку будь-якого бізнесу. Це забезпечить більшу відомість про вашу компанію і підвищить її довіру серед клієнтів. Це буде великою перевагою серед своїх конкурентів в даній сфері.

Основні переваги вебсайту для компанії:

- можливість працювати з клієнтами з будь якого куточку світу;
- економія грошей компанії на відкриття нових офісів в різних країнах;
- залучення нових клієнтів та можливість утримати старих;
- консультація клієнтів цілодобово.

Вебсайт для компаній – це перед усім обличчя бізнесу в інтернеті тому воно повинно бути таким щоб користувачі передали увагу саму йому а не суперникам. Інформація повинна бути інформативною і не дуже затягнутою.

Велике значення має сам дизайн вебсайту, тому що це найперше з чим стикається користувач. Якщо дизайн зроблений по всім сучасним тенденціям, це збільшить позитивне враження користувача від нашого вебсайту.

Головні аспекти гарного дизайну;

- візуально привабливий;
- виділити важливі моменти;
- адаптувати під мобільні користувачів;
- легка навігація;
- зміст повинен відповідати бренду компанії.

Адаптація під мобільні пристрої призведе до більшого об'єму користувачів.



Рисунок 1.1 – Поганий приклад дизайну вебсайту

На рисунку 1.1 зображений поганий дизайн, на якому погано підібрані кольори та візуально виглядає не дуже привабливим для користувача. А ось на рисунку 1.2 продуманий дизайн, бо має гармонійні кольори і привабливі шрифти створюють позитивне перше враження.

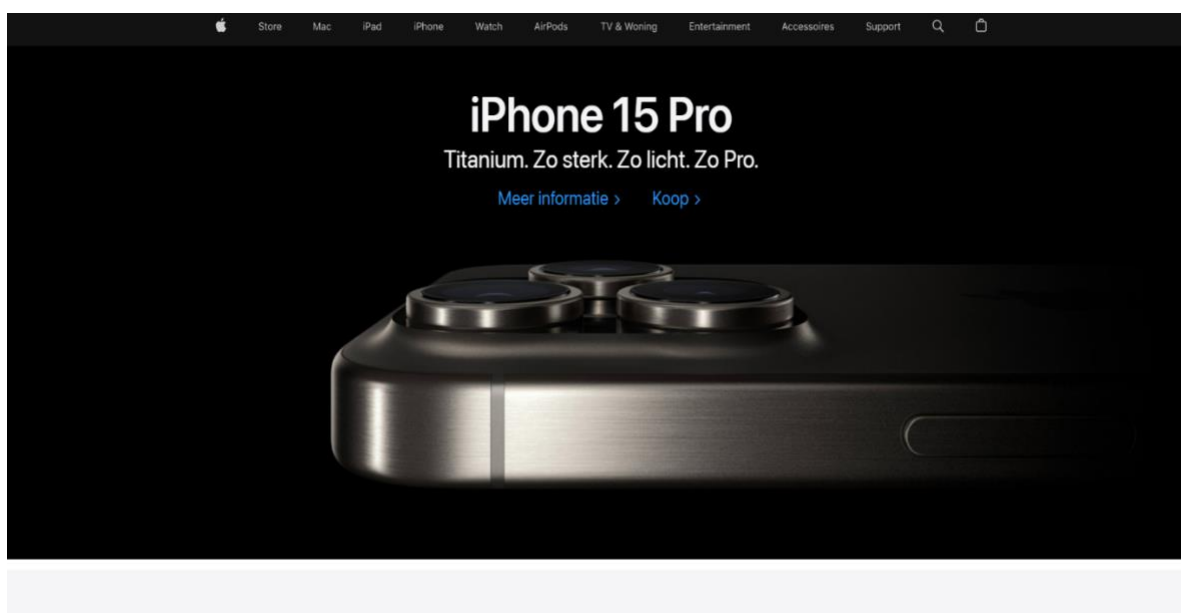


Рисунок 1.2 - Гарний приклад дизайну вебсайту

Якщо ваша компанія все ж вирішилась на цю велику можливість, то треба використовувати найбільш сучасні методи для розробки програмного забезпечення.

1.2 Дослідження методів для підвищення захисту вебсайту

Щороку збільшується кількість вебсайтів, які потрапляють під атаку. Як власник сайту, вам може бути неочевидно, що безпека вашого вебресурсу постійно під загрозою. Практично кожна компанія зберігає та обробляє особисті дані користувачів, а також конфіденційну інформацію та проводить онлайн-платежі.

Навіть якщо основною метою злому не є крадіжка даних або порушення роботи сайту, кіберзлочинці можуть мати інші мотиви, такі як відправка спаму або тимчасове використання вебсервера для зберігання файлів, часто з незаконним вмістом.

Зломи здійснюються спеціально створеними автоматизованими скриптами. Вони "переглядають" Інтернет у спробі зламати сайти, які мають відомі вразливості в програмному забезпеченні.

Навіть якщо ви є власником невеликого бізнесу, важливо не недооцінювати ризик кіберзлочинності. Існує значна ймовірність того, що ваш сайт може стати мішенню для зловмисників. Малий обсяг вебресурсу часто використовується шахраями для перевірки методів вторгнення та крадіжки даних, які потім можуть використовуватися для подальших атак на інші вебсайти.

Основні проблеми безпеки вебсайтів:

- використання HTTP замість HTTPS;
- злом паролів;
- викрадення даних бази даних.

Зазвичай основною причиною злому вебсайтів вважають небезпечну передачу даних. Дані передаються через стандартний протокол передачі даних -

HTTP. Однак цей протокол має свої вразливості, але їх можна виправити, перейшовши на протокол HTTPS.

Останній забезпечує захищену передачу даних, що дозволяє зберегти конфіденційність інформації при передачі від сервера користувачеві та навпаки. Для того, щоб перейти на роботу з протоколом HTTPS, потрібно отримати сертифікати безпеки для сайту. Навіть якщо використання HTTPS не є обов'язковим, якщо на вашому сайті передбачається використання платіжних систем, відчутно рекомендується використовувати його з кількох причин:

- користувачі частіше довіряють сайтам із протоколом HTTPS;
- HTTPS дає змогу захистити дані користувачів;
- вебсайти з цим протоколом найбільш підходять для політик браузерів.

Також одним з найбільш розповсюджених способів захисту вебсайтів, є хешування паролів. За допомогою цього зловмисники не зможуть перехватити паролі користувачів. За для цього використовується особливий алгоритм хешування.

Алгоритми хешування використовуються як односторонні функції, які беруть будь-який рядок і перетворюють його в унікальний "відбиток пальця" фіксованої довжини. Важливо відзначити, що цей процес неможливо розшифрувати назад до початкового рядка. Отже, в разі порушення безпеки бази даних зловмиснику важко або навіть неможливо отримати доступ до паролів користувачів, якщо вони були надійно хешовані.

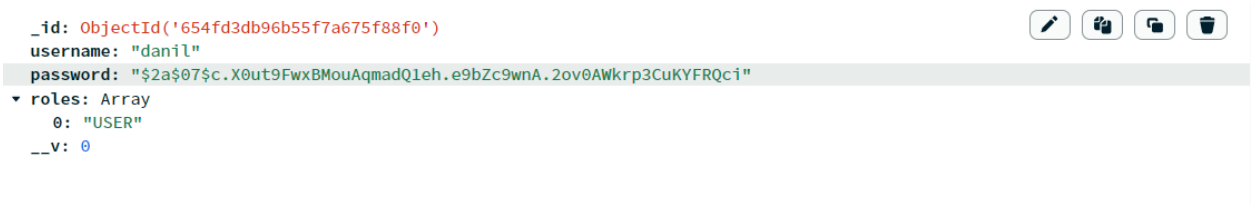
Хеш або хеш-функція - це криптографічний інструмент, який зазвичай називають просто "хеш". Хеш-функція представляє собою математичний алгоритм, що перетворює будь-який набір даних у рядок фіксованої довжини, що складається з цифр і літер. Основна ідея використовуваних у цьому контексті функцій полягає в використанні детермінованого алгоритму.

Це означає використання алгоритму, який завжди генерує унікальний та заздалегідь визначений результат при отриманні вхідних даних. Іншими словами, при одержанні однакових вхідних даних буде створюватися та сама рядок фіксованої довжини (використання однакового вводу завжди призводить до

отримання того ж результату). Детермінізм є ключовою властивістю цього алгоритму. Якщо змінити хоча б один символ в вхідних даних, буде створено абсолютно інший хеш.

```
_id: ObjectId('6532c8a38d727ffb937b719c')
username: "krosanter"
password: "$2a$07$mU6KrIydb7TnnT1Sgnrbx0SM6pck4Bjzq7vQwft/.YHZAhhrt3Jt."
roles: Array
  0: "ADMIN"
__v: 0
```

Рисунок 1.4 – Запис в базі даних користувача krosanter



```
_id: ObjectId('654fd3db96b55f7a675f88f0')
username: "danil"
password: "$2a$07$c.X0ut9FwxBMouAqmadQ1eh.e9bZc9wnA.2ov0AWkrp3CuKYFRQci"
roles: Array
  0: "USER"
__v: 0
```

Рисунок 1.5- Запис в базі даних користувача danil

На рисунках 1.4 та 1.5 зображені записи в базі даних користувача де пароль вже хешовані та записаний в базу даних. Ми можемо побачити наскільки вони різні.

Однією з проблем, які виникають в криптографічних хеш-функціях, є можливість виникнення колізій, тобто ситуацій, коли різні вхідні дані призводять до одного й того ж хеш-значення. Оскільки розмір хешу фіксований, існує можливість, що для різних вхідних даних може бути створений однаковий хеш. Це відкриває можливість для хакерів створювати колізії, тобто використовувати шкідливі дані, які мають той самий хеш, що й правильні дані.

Головна мета ефективних криптографічних хеш-функцій - максимально ускладнити можливість знаходження способів генерації вхідних даних, які дають однакове значення для хешування. Як вже було сказано раніше, обчислення хешу не повинно бути простим, а сам алгоритм повинен бути стійким до "атак на визначення початкових даних". Практично необхідно, щоб визначити обернені детерміновані кроки, які були вжиті для відтворення створеного хешем значення, було надзвичайно складно (а краще - неможливо).

Незалежно від використаних технологій шифрування та криптографічних інновацій у цьому напрямку, основним завданням є вирішення одного з двох питань:

- ускладнення внутрішніх операцій хешування;
- збільшення довжини вихідних даних хешу з урахуванням того, що обчислювальні можливості атакуючих не зможуть ефективно знаходити колізії.

Навіть при можливому розвитку квантових комп'ютерів у майбутньому експерти вважають, що традиційні інструменти, такі як хеш-функції, залишаться надійними. Це пояснюється тим, що зі зростанням обчислювальних можливостей математична структура внутрішніх алгоритмічних хеш-конструкцій стає менш вразливою. Квантові обчислення, в свою чергу, найбільше ефективні в розгляді об'єктів з чіткою математичною структурою.

1.3 Інтеграція чат-ботів в свій вебсайт

ChatGPT OpenAI - це чат-бот зі штучним інтелектом, який спілкується з користувачами, відповідаючи на запитання в інтерактивному форматі. Компанія запустила цей продукт 30 листопада 2022 року. Бот отримав широку популярність одразу після релізу завдяки своєму революційному прориву в галузі штучного інтелекту [1].

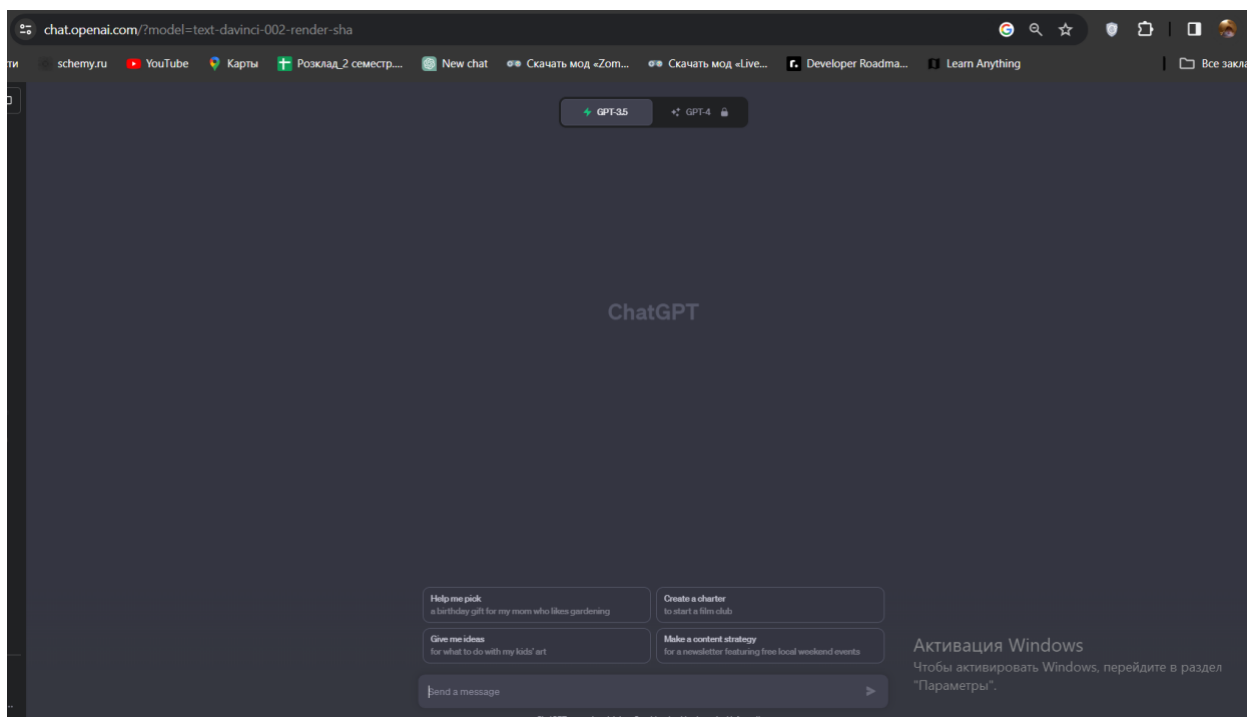


Рисунок 1.6- GhatGPT-3.5

Випущена у відкритий доступ версія може створювати основний програмний код, виробляти аналіз даних, публікувати узагальнені наукові статті та генерувати музику. Крім цього, бот вміє дотримуватися базових етичних стандартів, що сприяє позитивному сприйняттю аудиторії.

ChatGPT написаний на мові програмування Python. Бот ChatGPT ґрунтується на архітектурній моделі GPT-3.5 - удосконаленій версії GPT-3. Завдяки навчанню з учителем бот може аналізувати великі обсяги даних і робити прогнози. Мовна модель бота навчалася на великій кількості даних з різних текстів, матеріалів з Вікіпедії, публікацій у соціальних мережах тощо. Навчання відбувалося на двох основних етапах [1]:

- люди, які були користувачами, навчали ІІІ та в той же час були асистентами у створенні відповіді;
- люди оцінювали різні варіанти відповідей, написаних ІІІ, за актуальність, що дозволило йому навчитися визначати найточніший варіант відповіді;

– з ботом можна спілкуватися українською мовою, проте англomовні відповіді бот буде генерувати швидше.

Часто порівнюють чат-бота із звичайними пошуковими системами. Відмінність полягає в тому, що отримуючи запит, Google видає список найбільш релевантних посилань, а бот аналізує необхідну інформацію і видає її у вигляді одного тексту, уникнувши зайвих частин, про які не було зазначено в запиті.

Бот не просто відповідає на окремі запитання; він може брати участь у поточній розмові на основі попереднього введення користувача. Це дозволяє користувачеві надавати більше деталей і уточнювати свій запит, поки не отримає відповідь, яка повністю відповідає його очікуванням.

Реєстрація в чаті GPT займає кілька хвилин - для цього потрібно зайти на офіційний сайт OpenAI, ввести свою електронну адресу, мобільний номер телефону та ввести код, який прийде на номер. Після реєстрації користувач може починати користуватися чатом GPT в Україні [2].

На головній сторінці бота представлені різноманітні інформаційні блоки, наприклад приклади запитів, його можливості та обмеження. Незважаючи на широку популярність, користувачі іноді можуть отримувати повідомлення про помилки замість відповідей через обмеження одночасного вхідного трафіку чату GPT. У таких випадках користувачам пропонується спробувати ввести свій запит ще раз або повторно відвідати сайт пізніше.

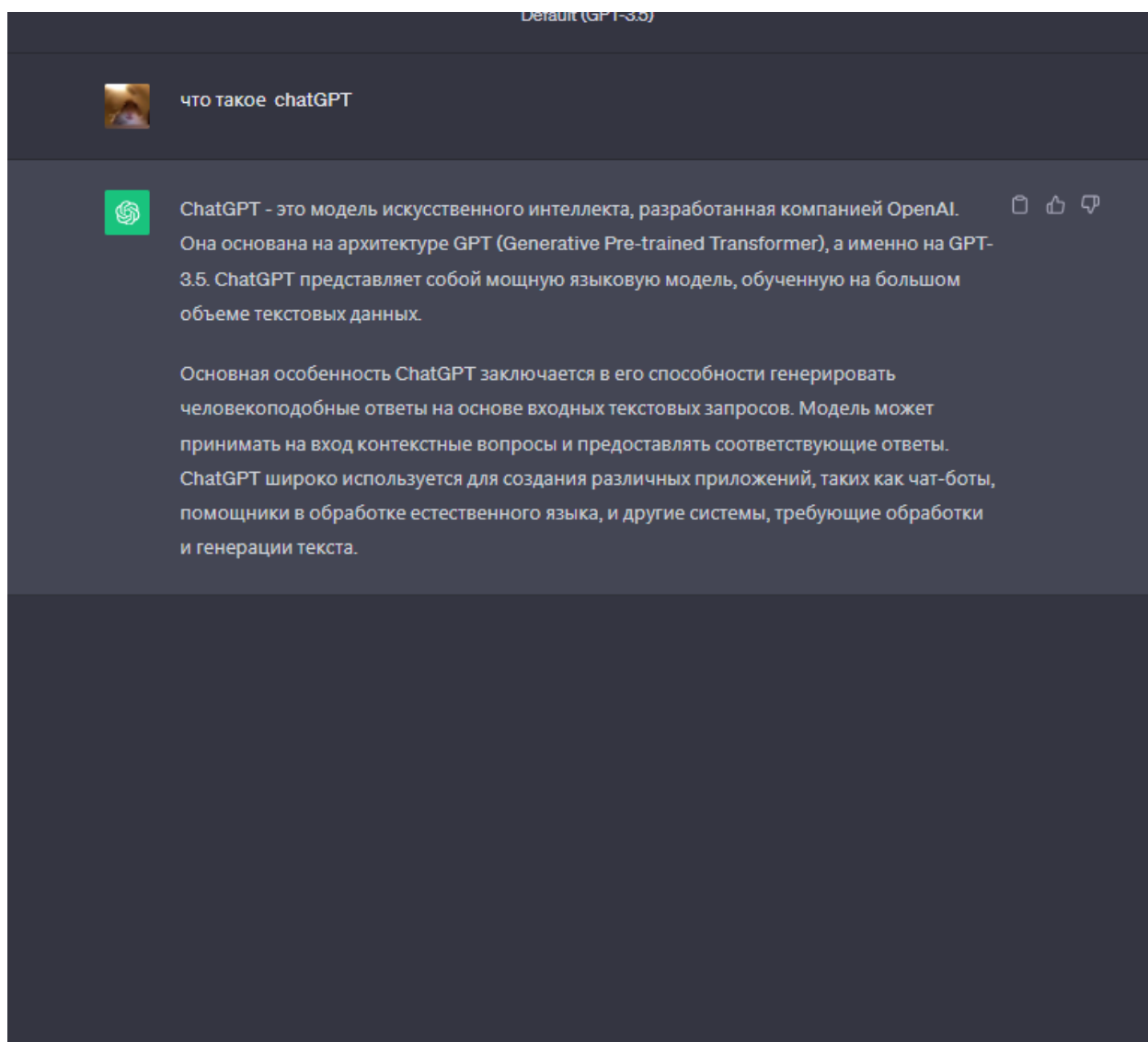


Рисунок 1.7 – Робота ChatGPT

GPT chat може виконувати безліч функцій:

- надання продуманих відповідей: бот може запропонувати добре продумані відповіді на широкий спектр питань, часто перевершуючи пошук Google за якістю. Користувачі цінують здатність бота надавати більш глибокі відповіді;
- допомога в кодуванні: GPT chat має навички перекладу коду між мовами програмування, пропонуючи вказівки щодо конкретних завдань кодування, розв'язуючи алгоритмічні проблеми, виявляючи помилки та навіть самостійно створюючи невеликі програми;
- вивільнення креативності: бот чудово створює сценарії, оповідання, есе, тексти пісень і нотні записи, демонструючи свої творчі здібності;

- обробка даних у електронних таблицях: у чаті GPT можна створювати таблиці з різними розрахунками чи планами, як-от розробляти режими харчування чи допомагати у фінансовому плануванні;
- співпраця з іншими нейронними мережами: бот може генерувати описи зображень, які є цінними для інших систем штучного інтелекту;
- можливості бота охоплюють широкий спектр, що ускладнює визначення конкретних запитів. Уявіть це як компаньйона в чаті, схожого на пошукову систему, яка розуміє не лише ключові слова, але й інші відомості, надані користувачем.

Варто зазначити, що хоча бот не завжди може мати негайну відповідь, він часто відповідає точно, коли користувачі перефразують свої запити. Щоб дізнатися більше про те, як користуватися чатом GPT з користю, слід визначитися з метою взаємодії. Більшість вмінь бота користувачі вивчають, тестуючи власні запитання та отримуючи результати. У випадку, якщо бот не надав всю необхідну інформацію через обмеження кількості символів, слід ввести команду "продовжити останню відповідь" чи подібну. Бот не буде дублювати попереднє повідомлення, а надати додаткові дані.

Недоліки ChatGPT:

- неможливість створення інформації про сучасні події, оскільки бот не має доступу до даних, що відбулися після 2021 року;
- визначеність контенту, створеного ботом, наприклад, використання чат-бота для створення SEO-контенту може погіршити рейтинг сайту в пошукових системах через визначення його як штучно створеного;
- можливість отримання неправдивої інформації через те, що прототип ChatGPT створений для виявлення недоліків, і деякі відповіді можуть бути неточними;
- занепокоєння фахівців у галузі освіти стосовно використання бота, оскільки він може допомагати учням уникнути самостійного виконання завдань, що може впливати на оцінювання їхніх знань.

Нейронні мережі та штучний інтелект активно прогресують, еволюціонуючи разом зі своїм зростаючим впливом на суспільство. Новітні технології розширюють можливості людства, автоматизуючи повсякденні завдання та пропонуючи консультації, підтримку та допомогу. Інформація Всесвітнього економічного форуму свідчить про те, що в майбутньому штучний інтелект зможе автоматизувати 85 мільйонів робочих місць. Однак, як очікується, одночасний розвиток нових технологій створить 95 мільйонів нових робочих місць.

Для ознайомлення з можливостями чату GPT та розширення знань щодо його функціоналу, важливо провести власне тестування. Варто пам'ятати, що необхідно оцінювати інформацію, отриману від бота, критично, і використовувати інші джерела для пошуку та обробки інформації.[3]

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Вимоги для функціоналу системи

Для даного дипломного проєкту буде розроблено вебсайт, який виступатиме в ролі інформаційного центру бібліотеки. Користувачі матимуть змогу здобувати інформацію про різноманітні книги, дані про які будуть зберігатися в базі даних.

Вебсайт повинен надати функцію реєстрації та авторизації для отримання особливих функцій. Також треба надати особливий доступ для адміністраторів які будуть змінювати та редагувати контент. Для цього повинна буде реалізована система ролей.

Не треба забувати про те, що користувачі повинні отримувати допомогу від інтелектуального помічника. Доступ до якого користувачі отримають тільки після авторизації.

Необхідно подбати про захист системи. Наприклад, буде реалізована система хешування паролів. Це захистить від злочинців, які захочуть заволодіти даними

користувачів. Також буде використовуватися система jwt токенів як електронний підпис користувача.

2.2 Розробка структурної схеми

Структурна схема необхідна для опису основної структури вебсайту. Взаємозв'язок між сторінками нашої системи та основної інформації про неї.

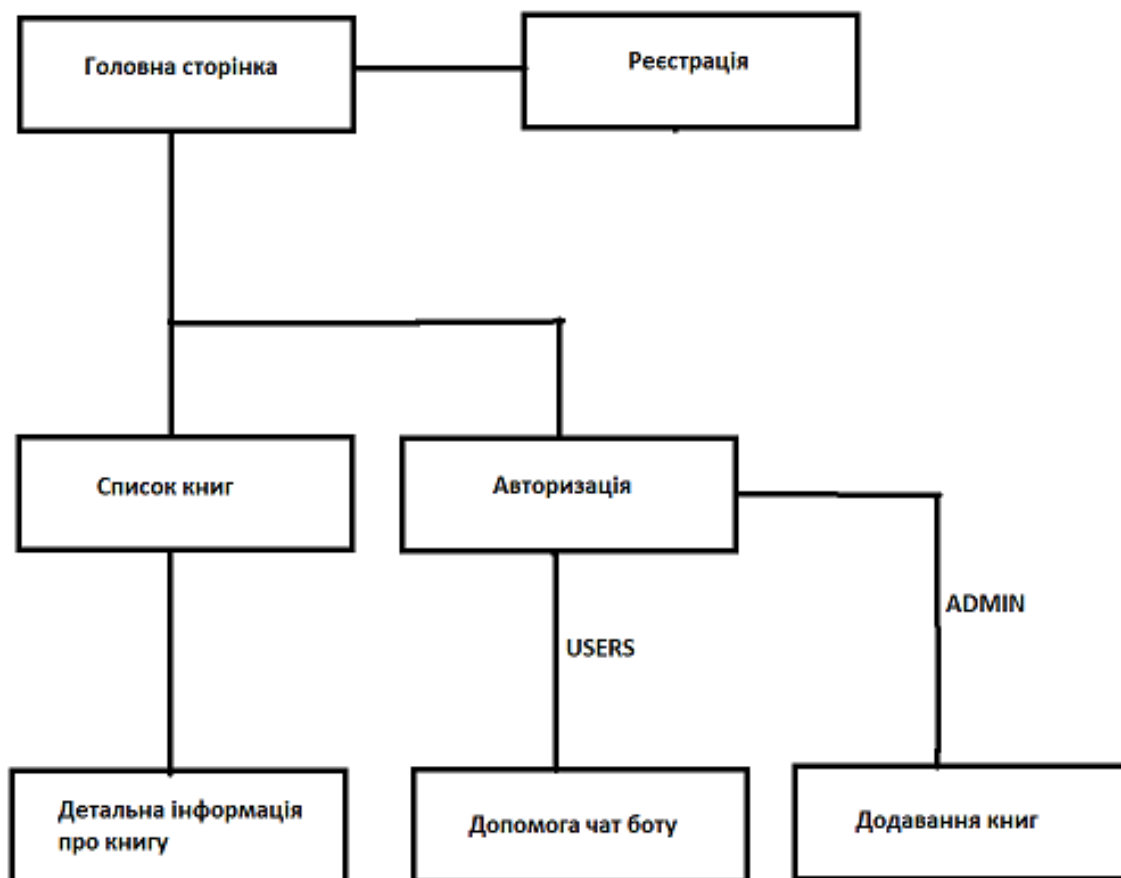


Рисунок 2.1- Структурна система вебсайту

На рис. 2.1 зображено структурну схему, яка описує основну структура розробленого вебсайту.

На головній сторінці описується загальна інформація про вебсайт, зображено основну інформацію. Ця сторінка – перше, що бачить користувач, коли заходить на вебсайт.

На сторінці Реєстрації користувачі можуть зареєструватися на вебсайти. Для цього потрібно заповнити форми логін та пароль, для того щоб вони збереглись в базі даних.

На сторінці Список книг знаходяться список книг які представлені на вебсайті та які зберігаються в базі даних. В списку записані основні дані про книги назва, Рік випуску та автора.

На сторінці Детальна інформація про книгу записана інформація про книгу, яку захотів дізнатися користувач. Дані надаються за id книги, яку захотів користувач.

На сторінці Авторизація користувачі можуть авторизуватися під різні ролі, наприклад як USERS або ADMIN. Якщо авторизуватися як USERS можна отримати доступ до допомоги інтелектуального чат-боту. Користувачі які авторизувались як ADMIN отримають доступ до додавання книг до розробленого вебсайту.

На сторінці Допомога чат-боту користувачі зможуть отримати допомогу інтелектуальних чат-ботів.

На сторінці Додавання книг адміністратори зможуть додавати книги до бази даних.

2.3Вибір технологій для розробки

2.3.1 Мова програмування JavaScript

Раніше інтерфейс сайтів складався з текстів, посилань і ілюстрацій, а верстали його за допомогою мов HTML і CSS.

Все змінилося зі створенням JavaScript. Цей мову програмування зроблено сайти інтерактивними — це зручно для користувача. Появились кнопки, форми, поля вводу і багато іншого.

JavaScript — це багатофункціональний мову програмування. JavaScript використовується не тільки для створення сайтів, але і для опису дуже різних програм.

За допомогою JavaScript розробники створюють вебприкладення — це програми, які ми запускаємо в браузері. До нього відносяться поштові клієнти, редактори тексту, соціальні мережі, відеохостинг і багато іншого [4].



Рисунок 2.2- Популярність мов програмувань для веброзробки

На JavaScript створюють «зовнішній інтерфейс» вебзастосунків — фронтенд. З ним взаємодіє користувач, натискаючи на кнопки, плашки, меню.

На JavaScript написаний код для популярних сервісів у Google Maps, Google Docs, Netflix, eBay.

Крім інтерфейсу, який бачать користувачі, у вебзастосунках є і внутрішня частина — бекенд. Це серверна частина додатків, у якій зазвичай зберігаються бази даних.

У JavaScript є Node.js — це не окрема мова програмування, середовище виконання JavaScript-коду на стороні сервера. З її допомогою можна виконати операцію або звернутися до бази даних.

JavaScript у зв'язку з Node.js використовують відомі компанії PayPal і Walmart для взаємодії зі своїми серверами.

Навколо JavaScript була сформована величезна екосистема бібліотек і фреймворків, таких як React, Angular і Vue.js. Ці інструменти роблять розробку більш ефективною.

Володіння ними — обов'язкова вимога, яке є у багатьох вакансіях.

Зараз активно розвивається і платформа Node.js, за допомогою якої, як ми вже розібралися, можна писати серверні додатки. Так що бекенд на JavaScript також буде відновлений.

2.3.2 Середовище виконання Node.js

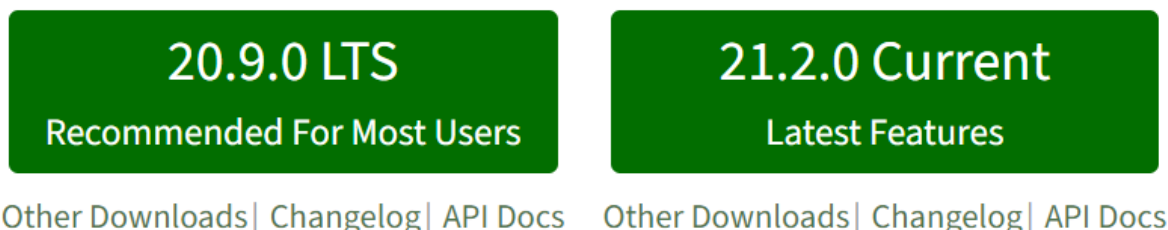
Node.js — це середовище виконання коду JavaScript поза браузером, яке дозволяє писати серверний код для вебсторінок і вебдодатків, а також для командних рядків програми.

Node.js — не окрема мова програмування, платформа для використання JavaScript на стороні сервера. За допомогою платформи можна працювати з файлами, мережами, базами даних та іншими системними ресурсами на сервері [5].

Якщо говорити про мову, то як для фронтенду, так і для бекенда використовується один і той же JavaScript. Різниця тільки в наборі API, які використовують фронтендери та бекендери. Браузерний JavaScript використовує вебінтерфейс API, який забезпечує доступ до DOM і користувацького інтерфейсу сторінок. Серверний JavaScript використовує API, забезпечує доступ до файлової системи додатків, HTTP-запитів, потоків[6].

Node.js® is an open-source, cross-platform JavaScript runtime environment.

Download Node.js®



For information about supported releases, see the [release schedule](#).

Рисунок 2.3 - Встановлення Node.js

Платформа Node.js була представлена в 2009 році — її створив інженер Райан Дал, а спонсором розробки виступила компанія Joyent. Вона відома підтримкою open-сурсних проєктів, включаючи Node.js, Illumos, SmartOS.

Для створення Node.js Райан Дал використовував двигун V8. Платформа реалізована з низькорівневою неблокуючою моделлю вводу/виводу, яка побудована на сюжетно-орієнтованій моделі. Це означає, що Node.js може обробляти кілька запитів одночасно, без блокування інших операцій.

В кінці 2014 року інженер Федір Індутний, який увійшов в основну команду розробників платформи, створив популярний фреймворк Node.js. Фреймворк з'явився із-за неприємностей розробників політики компанії Joyent.

Платформа io.js перевершила Node.js у продуктивності, але творці фреймворка вже в 2015 році вирішили об'єднатися з Node.js, щоб вплинути на розвиток основної платформи. В даний час розробкою офіційно керує Node.js Foundation [7].

2.3.3 Мова розмітки HTML та стилізації CSS

Стандарт HTML є мовою подання гіпертексту, запропонованою організацією W3C (World Wide Web Consortium) і розробленою на початку 1996 року в кооперації з такими фірмами як IBM, Microsoft, Netscape Communication

Corporation, Novell, SoftQuad, Spyglass та Sun Microsystems. Організація W3C спільно із згаданими фірмами продовжує роботу над розширенням можливостей елементів мови HTML і інтернаціоналізацією мови. Основні розділи документу HTML Документ HTML можна розглядати як сукупність вказівок (команд) і даних (як безпосередньо розміщених у документів, так і пов'язаних з ним посиланнями), які при інтерпретації програмою-браузером відтворюють вигляд сторінок документу. Документ HTML складається з так званих тегів [8].

Web-сторінки можуть існувати в будь-якому форматі, але в Як стандарт прийнятий Hyper Text Markup Language - мова розмітки гіпертекстів, призначений для створення форматowanego тексту, насиченого зображеннями, звуком, анімацією, відео та гіпертекстовими посиланнями на інші документи.

Можна працювати на Web без знання мови HTML, оскільки тексти HTML можуть створюватися різними спеціальними редакторами і конвертерами. Але писати безпосередньо на HTML неважко. Можливо, це навіть легше, ніж вивчати HTML-редактор або конвертер, які часто обмежені в своїх можливостях, містять помилки або проводять поганий HTML код, який не працює на різних платформах.

Мова HTML існує в декількох варіантах і продовжує розвиватися, але конструкції HTML швидше за все будуть використовуватися і в надалі. Вивчаючи HTML і пізнаючи його глибше, створюючи документ на початку вивчення HTML і розширюючи його наскільки це можливо, ми маємо можливість створювати Web-сторінки, які можуть бути переглянуті багатьма браузерами Web, як зараз, так і в майбутньому. Це не виключає можливості використання інших методів, наприклад, метод розширених можливостей, що надається Opera, Google Chrome, Internet Explorer або іншими браузерами. Робота по HTML - це спосіб засвоїти особливості створення документів у стандартизованій мові, використовуючи розширення, тільки якщо це дійсно необхідно.

HTML був ратифікований World Wide Web Consortium. Він підтримується всіма браузерами.

Оскільки HTML-документи записуються в ASCII I-форматі, то для її створення може використаний будь-який текстовий редактор.

Зазвичай HTML-документ - це файл з розширенням. html або htm, в якому текст розмічений HTML-тегами (англ. tag - спеціальні вбудовані вказівки). Засобами HTML задаються синтаксис і розміщення тегів, в відповідно до яких браузер відображає вміст Вебдокумента. Текст самих тегів Веббраузером не відображається[9].

Всі теги починаються символом '<' і закінчуються символом '>'. Зазвичай є пара тегів - стартовий (відкриває) і завершальний (Закриває) тег (схоже на що відкриваються і закриваються дужки в математиці), між якими міститься розмічається інформація

Під терміном CSS ми розуміємо фразу "Каскадні таблиці стилів". Вони впроваджуються в HTML-код і не вимагають жодних спеціальних редакторів і компіляторів, адже інтерпретуються звичайним браузером.

Каскадні таблиці стилів є описами різних HTML-елементів і створені вони для розширення властивостей останніх. Вперше стилі було запропоновано WWW Consortium'ом у рамках розробки специфікації HTML 3.0, проте реально цю шнягу почали підтримувати лише 1997 року. Насолодитися CSS мали можливість лише щасливі власники таких браузерів, як Netscape Navigator 4.0 та Internet Explorer 4.0 [10].

На цьому з історичними фактами дозвольте зав'язати та розпочати безпосередньо справу. CSS відкриває нам нові, раніше невідомі та недоступні грані. З їхньою допомогою ми отримуємо можливість більш вишукано оформити свій вебсайт, надати йому нових рис та симпатичної особи [11].

Синтаксис CSS досить простий у вивченні, тому освоєння каскадних стилів є досить легкою, але дуже корисною. Для того щоб не виявитися голослівним, я хочу навести невеликий приклад. Візьмемо типову таблицю HTML та поставимо атрибут border="1". Після інтерпретації цього коду броузером ми отримаємо звичайнісінький результат, до якого звик наше досвідчене око. Інша справа, що таблиці подібного роду не сумісні з жодним більш-менш хорошим дизайном. Ну куди годиться ця псевдо-тривимірна лінія, що є кордоном? Ось тут і приходять на

допомогу CSS, за допомогою яких можна без зайвих проблем оформити будь-яку таблицю на власний смак .

2.3.4 Інструменти для покращення вебсайт

Mongoose – надає у розпорядження розробників просте засноване на схемах рішення для моделювання даних програми, що включає вбудовану перевірку типів, валідацію, формування запитів та хуки, які відповідають за реалізацію додаткової логіки обробки запитів [12].

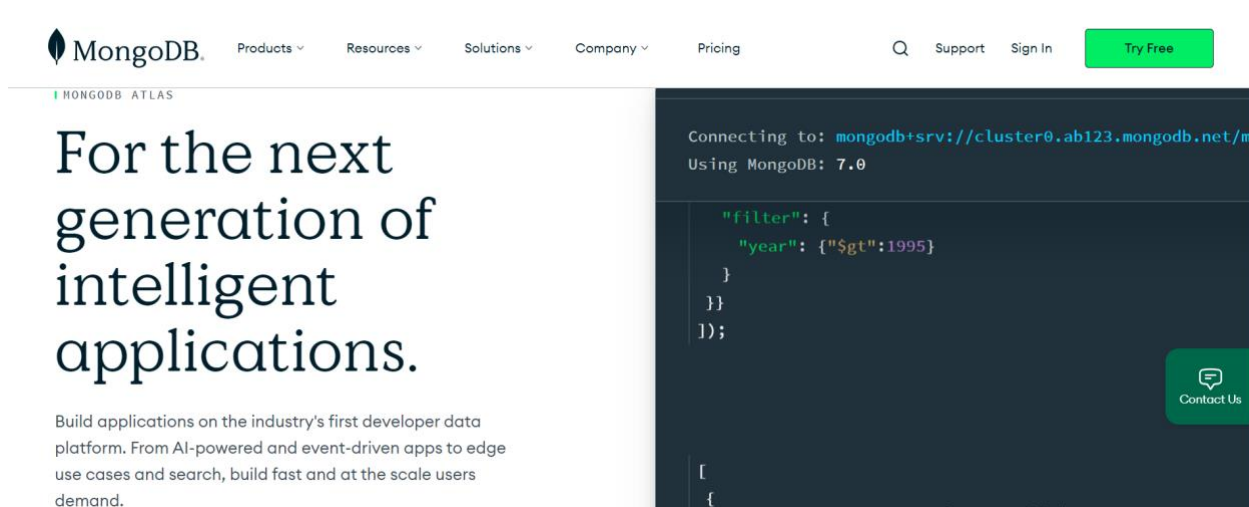


Рисунок 2.4 - MongoDB

Mongoose надає величезний набір функціональних можливостей для створення та роботи зі схемами. На даний момент Mongoose містить вісім SchemaTypes (* типи даних схеми), які можуть мати властивість, що зберігається в MongoDB. Ці типи такі:

- String;
- Номер;
- Date;
- Buffer;
- Boolean;
- Mixed;
- ObjectId (* унікальний ідентифікатор об'єкта, первинний ключ, _id);
- Array.

Для кожного типу даних можна:

- встановити значення за замовчуванням;
- задати користувальницьку функцію перевірки даних;
- вказати, що поле необхідно заповнити;
- задати `get`-функцію (гетер), яка дозволяє вам проводити маніпуляції з даними до повернення у вигляді об'єкта;
- встановити `set`-функцію, яка дозволяє вам проводити маніпуляції з даними до їх збереження в базу даних;
- визначити індекси для швидшого отримання даних.

Крім цих загальних можливостей для деяких типів даних також можна налаштувати особливості збереження та отримання даних із бази даних. Наприклад, для типу даних `String` можна вказати такі додаткові опції:

- конвертація даних у нижній регістр;
- конвертація даних у верхній регістр;
- обрізання даних перед збереженням;
- визначення регулярного виразу, що дозволяє в процесі перевірки даних;
- обмежити дозволені для збереження варіанти даних;
- визначення переліку, який дозволяє встановити список допустимих рядків.

Express JS — це платформа для запуску програм, яка використовується для створення вебдодатків та сторінок сайтів. Використання Express JS значно пришвидшує процедуру створення сайтів та дає змогу розробникам розширити перелік доступних їм інструментів [13].

Низький поріг входу до застосування фреймворку не означає, що ви зможете користуватися всіма його функціями, але більшість простих можливостей буде вам доступна з самого початку.

Handlebars - це шаблонізатор, заснований на Mustache, із додатковими можливостями. Він використовує подвійні фігурні дужки для вставки змінних та

блоків коду. У цьому прикладі `{{title}}` , `{{header}}` та `{{content}}` замінюються відповідними значеннями при обробці шаблону [14].

JSON Web Token (JWT) містить три блоки, розділених точками: заголовок (header), набір полів (payload) і сигнатуру. Перші два блоки представлені у JSON-форматі та додатково закодовані у формат base64. Набір полів містить довільні пари ім'я/значення, стандарт JWT визначає кілька зарезервованих імен (iss, aud, exp та інші). Сигнатура може генеруватися за допомогою симетричних алгоритмів шифрування, і асиметричних. Крім того, існує окремий стандарт, що відписує формат зашифрованого JWT-токена [15].

Токени є засобом авторизації для кожного запиту від клієнта до сервера. Токени (і відповідно сигнатура токена) генеруються на сервері ґрунтуючись на секретному ключі (який зберігається на сервері) та payload'е. Токен у результаті зберігається на клієнта і використовується за необхідності авторизації будь-якого запиту. Таке рішення відмінно підходить для розробки SPA.

При спробі хакером підмінити дані в header'і або payload'і, токен стане не валідним, оскільки сигнатура не відповідатиме початковим значенням. А можливість згенерувати нову сигнатуру у хакера відсутня, оскільки секретний ключ для зашифрування лежить на сервері.

Bcrypt.js є бібліотекою, розробленою для забезпечення безпеки зберігання паролів у вебдодатках. Ця бібліотека є реалізацією алгоритму хешування паролів Bcrypt, широко визнаного в індустрії інформаційної безпеки завдяки своїй стійкості до атак підбору [16].

Bcrypt (Blowfish Crypt) — це криптографічна функція хешування, розроблена для забезпечення безпечного зберігання паролів. Bcrypt використовує адаптивний алгоритм хешування, що робить його особливо ефективним проти атак підбору з використанням графічних процесорів (GPU) або спеціалізованих ASIC-пристроїв[17].

Bcrypt.js надає зручний інтерфейс для використання алгоритму Bcrypt у вебдодатках. Деякі ключові особливості бібліотеки включають:

- простота використання: Bcrypt.js надає прості методи для хешування паролів та порівняння хешів, що спрощує інтеграцію до існуючого коду;
- налаштування параметрів: Бібліотека дозволяє налаштовувати параметри хешування, такі як вартість, що забезпечує гнучкість у виборі рівня безпеки;
- підтримка асинхронності: Bcrypt.js підтримує асинхронні операції, що корисно у вебзастосунках, де обробка паролів може займати значний час.

3 РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1 Реалізація серверної частини

3.1.1 Головний файл вебсайту Index.js

Index.js являє собою головний файл проєкту. В цьому файлі відбувається ініціалізація, підключення залежностей і запуск основної логіки програми.

В лістингу 3.1 описано як на початку файлу підключення основних залежностей додатку які будуть використовуватися далі.

Лістинг 3.1 – підключення різних частин до index.js

```
const express = require ('express')
const exphbs = require ('express-handlebars')
const mongoose = require ('mongoose')
const authRouter = require('./authRouter')
const bodyParser = require('body-parser')
const BookHaven = require('./route/BookHaven')
const routes = require('./routes')
const openai = require('openai');
```

У лістингу 3.2 Встановлюються 2 важливих зміни PORT та apiKey.PORT- дає значення зміненого порту, на який буде запущено веб-сайт. apiKey- представляє собою ключ для доступу до ChatGPT.

Лістинг 3.2 – Встановлення змінних PORT , api Key

```
const PORT = process.env.PORT || 5000
const apiKey = 'sk-
BIAHJepX4vnqhnG3A8vhT3BlbkFJEVBB0nvgD5z73g9uHhcB';
```

У лістингу 3.3 створюється примірник Express-програми яка буде використовуватися для обробки HTTP запитів і створення вебсервера. Також у цьому лістингу задаються основні параметри для express-handlebars.

Лістинг 3.3 – Основні параметри Express та handlebars

```
const app = express()
const hbs = exphbs.create ({
  defaultLayout: 'main' ,
  extname: 'hbs'
})
```

В лістингу 3.4 налаштовується движок и механізм представлення для Express. В даному вебсайті буде використовуватися Handlebars. Також встановлюється директорію в якій знаходяться файли для рендеренгу. Вони знаходяться в директорії views.

Лістинг 3.4 – Налаштування express-handlebars

```
app.engine('hbs', hbs.engine)
app.set('view engine', 'hbs')
app.set('views', 'views')
```

В лістингу 3.5 підключаються основні middleware які будуть використовуватися в даному застосунку. App.use(bodyParser.urlencoded ({extended: true})) - ця частина Bodyparser яка відповідає за обробку даних з тіла HTTP запитів. app.use(bodyParser.json()) – ця частина відповідає за обробку даних у форматі JSON. Також підключає основні маршрути для нашого додатку для аутентифікації, навігації по вебсату.

Лістинг 3.5 – підключення middleware

```
app.use(bodyParser.urlencoded({ extended: true }))
app.use(bodyParser.json())
```

```
app.use(BookHaven)

app.use('/', routes)
app.use(express.json())
app.use("/auth", authRouter)
```

Функція яка наведена у лістингу 3.6 відповідає за запуск вебдодатку та підключення до бази даних MongoDB. Підключення до бази даних відбувається за допомогою особливого ключа який наведений в даному прикладі. Якщо з базою даних не вийде встановити зв'язок то наш додаток не буде запущений та виведе в консоль помилку. Як тільки буде встановлено стабільний зв'язок наш додаток запуститься на порту який ми задали раніше.

Лістинг 3.6 – Функція запуску додатку

```
const start = async () => {
  try {
    await
    mongoose.connect('mongodb+srv://danilbilash:krosanter@cluster0.
c20djhc.mongodb.net/?retryWrites=true&w=majority')
    app.listen(PORT, () => console.log (`server started on port
${PORT}`))
  } catch (e) {
    console.log(e)
  }
}
```

3.1.2 Реалізація маршрутизації

Routes.js- ця частина відповідає маршрутизацію основних маршрутів в нашому додатку. В цьому файлі будуть визначатися основні маршрути HTTP запитів.

В даному лістингу 3.7 імпортуються основні елементи які будуть використовуватися в даному файлі. Express.Router()- створення об'єкту маршрутизації в рамках Express.

Лістинг 3.7 – Імпорт елементів для даного файлу

```
const express = require('express');
const router = express.Router();
const Book = require('./models/books');
```

Лістинг 3.8 представляє собою GET запит для отримання списку книг які знаходяться в базі даних та рендерить їх на сторінці. Книги витягнуто з бази даних за допомогою команди `Book.find()`.

Лістинг 3.8 – Маршрут до /books

```
router.get('/books', async (req, res) => {
  try {
    const books = await Book.find();
    const sanitizedBooks = books.map(book => {
      return {
        _id: book.id,
        _title: book.title,
        _author: book.author,
        _description: book.description,
        id: book.id,
        title: book.title,
        author: book.author,
        description: book.description,
      };
    });

    res.render('books', { books: sanitizedBooks });
  } catch (err) {
    console.error('Error fetching books:', err);
    res.status(500).send('Internal Server Error');
  }
});
```

В лістингу 3.9 описується маршрути до форми додавання книжок, також до файлу який додає ці додає ці книги до бази даних.

Лістинг 3.9 – Маршрути до різних частин додатків

```
router.get('/add-book', (req, res) => {
  res.render('add_book');
});

router.get('/ask', (req, res) => {
  res.render('ask');
});

router.post('/add-book', async (req, res) => {
  try {
    const { title, author, description } = req.body;
    const newBook = new Book({ title, author, description });
    await newBook.save();
    res.redirect('/');
  }
});
```

```

    } catch (error) {
      console.error(error);
      res.status(500).send('Internal Server Error');
    }
  });

router.get('/books/:id', async (req, res) => {
  try {
    console.log (req.params.id)
    const books = await Book.findById(req.params.id);

    if (!books) {
      return res.render('bookDetails', { message: 'Book not
found' });
    }
  }
});

```

У лістингу 3.10 відбувається рендер сторінки де буде детальний опис книжки яку вибере користувач із списку. Дані будуть взяті із бази даних. Якщо виникнуть якісь проблеми то ця функція виведе помилку в консоль додатку.

Лістинг 3.10 – Отримання деталей про книгу

```

const sanitizedBook = {

  _title: books.title,
  _author: books.author,
  _description: books.description,
  title: books.title,
  author: books.author,
  description: books.description,
};

res.render('bookDetails', { books :sanitizedBook });
} catch (err) {
  console.error('Error fetching book details:', err);
  res.status(500).send('Internal Server Error');
}

});

module.exports = router;

```

3.1.3 Реалізація функцій реєстрації та авторизації

Ця частина відіграє головну роль для реєстрації та авторизації користувачів в даному проєкті. Так як в даному файлі описані основні функції для реєстрації та

аутентифікації, також в даному файлі шифруються паролі для зберігання в базі даних та створення JWT токенів.

Для хешування паролів використовується бібліотека `bcrypt` яка виконує його в 3 етапи:

- створення (SALT)- генерується випадкове слово яке буде знаходитися перед хешованим паролем для того щоб зменшити ефективність атак за допомогою(rainbow tables);
- ітерації – дана бібліотека дозволяє вказати з якою кількістю ітерацій повинно відбуватися хешування. Чим більше тим складніше буде розшифрувати сам хеш;
- хешування - на даній частині відбувається додавання (salt) в початок паролю та потім прогоняється через багато ітерацій для отримання унікального хешу.

Також в даній частині проєкту відбувається перевірка даних які вводять користувачі на валідність даних.

Лістинг 3.11 представляє собою інструкції для імпорту різних модулів та моделей для даної частини проєкту, які пов'язані з реєстрацією та аутентифікацією користувачів

Лістинг 3.11 – Імпорт різних модулів та моделей

```
const User = require('./models/user')
const Role = require('./models/Role')
const bcrypt = require('bcryptjs')
const jwt = require('jsonwebtoken')
const {validationResult} = require('express-validator')
const {secret} = require('./config')
```

Для створення jwt токена використовується функція наведена в лістингу 3.12. Для створення токена в функцію передається `id` та `roles` для генерації jwt токена. Для даного додатку токен буде мати цикл 24 години.

Лістинг 3.12 – Створення jwt токена

```
const generateAccessToken = (id, roles) => {

    const payload = {
        id,
        roles
    }

    return jwt.sign(payload, secret, {expiresIn: "24h"})
}
```

Лістинг 3.13 реалізує метод registration на нашому вебсайті. Даний етап включає певні етапи:

- перевіряє дані з форми які ввели користувачі на валідацію;
- перевіряє існує користувач з таким іменем в базі даних. Якщо введене ім'я повторюються виводиться помилка;
- хешується пароль за допомогою методу bcrypt;
- знаходить роль користувача в базі даних;
- створюється новий файл з ім'ям користувача, хешованим паролем та роль користувача та записує його в базу даних;
- повертає повідомлення про успішну реєстрацію.

Лістинг 3.13 –Реєстрація користувачів

```
class authController {
  async registration (req, res) {
    try {
      const errors = validationResult(req)
      if(!errors.isEmpty()) {
        return res.status(400).json({message: "Помилка реєстрації", errors})
      }
      const {username, password} = req.body
      const candidate =await User.findOne({username})
      if (candidate) {
        return res.status(400).json({message: " Користувач із таким ім'ям існує"})
      }
      const hashPassword = bcrypt.hashSync(password, 7);

      const userRole = await Role.findOne({value: "USER"})
```

```

    const user = new User({username, password: hashPassword,
roles: [userRole.value]})
    await user.save()
    return res.json ({message:" Користувач зареєстрований "})

  } catch (e) {
    console.log(e)
    res.status(400).json({message: 'Помилка реєстрації '})
  }
}

```

Для реалізації авторизації була створена функція функцію Login (лістинг 3.14) яка виконується в пару етапів:

- отримує дані від форми в яку вводять дані користувачі;
- пошук користувача в базі даних за ім'ям;
- перевіряє існує даний користувач в базі даних. Якщо ні то повертає помилку;
- перевірка пароля з бази даних та введеним користувачем. Якщо паролі не співпадають то повертає помилку;
- генерація токена доступу за вищеописаною функцією;
- повернення токена.

Лістинг 3.14 –Авторизація користувача

```

async login (req,res) {
  try {
    const {username, password} = req.body
    const user = await User.findOne({username})
    if (!user){

      return res.status(400).json({message: 'Користувача не
знайдено'})
    }
    const validPassword = bcrypt.compareSync (password,
user.password)
    if (!validPassword){
      return res.status(400).json({message: 'Неправильний
пароль '})
    }

    const token = generateAccessToken(user._id, user.roles)
    return res.json({token})
  }
}

```

```

    } catch (e) {
      console.log(e)
      res.status(400).json({message: ' Помилка реєстрації '})
    }
  }
}

```

В лістингу 3.15 наведена функція для отримання всіх користувачів з бази даних та передає їх в інші функції.

Лістинг 3.15 –отримання користувачів з бази даних

```

async getUsers (req, res) {
  try {
    const users = await User.find()
    res.json(users)

    res.json("server work")
  } catch (e) {
  }
}
}
module.exports = new authController()

```

3.1.4 Налаштування маршрутів для авторизації

В даному файлі відбувається налаштування маршрутів для автентифікації та авторизації користувачів в вашому вебдодатку .

Лістинг 3.16 відбувається імпорт для даного файлу різних елементів для налаштування маршрутів. Тут ми використовуємо фреймворк Express для створення екзмпляра Router. Об'єкт router стає контейнером для маршрутів та служить для обробки HTTP-запитів на стороні сервера.

Лістинг 3.16 – підключення різних частин для цього файлу

```

const Router =require ('express')
const router= new Router()
const controller = require ('./authController')
const {check} = require ("express-validator")
const authMiddleware =require ('./middleware/authMiddleware')
const roleMiddleware =require ('./middleware/roleMiddleware')

```

В лістингу 3.17 відбувається налаштування маршрутів :

- /registration POST-маршрут: При отриманні POST-запиту за адресою /registration, викликається метод controller.registration. Використовує функції валідації для перевірки введених даних;
- /login POST-маршрут: При отриманні POST-запиту за адресою /login, викликається метод controller.login;
- /users GET-маршрут: При отриманні GET-запиту за адресою /users, викликається метод controller.getUsers. Також використовує roleMiddleware для перевірки ролей користувача.

Лістинг 3.17 – Налаштування маршрутів

```
router.post('/registration', [
  check('username', " ім'я користувача не може бути пустим "
).notEmpty(),
  check('password', " пароль має бути більше 4 і менше 10
символів ").isLength({min:4,max:10})
], controller.registration)
router.post('/login', controller.login)
router.get('/users', roleMiddleware( ['ADMIN'] ),
controller.getUsers)

module.exports = router
```

3.1.6 Створення маршрутів для вебсайту BookHaven.js

Даний файл виконує маршрутизацію до різних сторінок вебсайту. Таких як головна сторінка , Вхід , Реєстрація.

В даному фрагменту (лістинг 3.18) коду наведено створення маршрутів для головних сторінок вебсату. Дані маршрути знаходяться в павбар нашого додатку.

Лістинг 3.18 – створення маршрутів

```
const { Router } = require('express')
const router = Router()

router.get('/', (req, res) => {
  res.render('index', {
    title : 'BookHaven'
  })
})
```

```

}))
router.get('/login', (req, res) =>{
  res.render('login',{
    title : 'Вхід'
  })
})

router.get('/regist', (req, res) =>{
  res.render('regist',{
    title : 'Реєстрація'
  })
})

module.exports = router

```

3.1.7 Моделі для роботи з базою даних

Моделі в бібліотеці Mongoose визначають структуру та правила для об'єктів, які будуть зберігатися в базі даних MongoDB. Вони дозволяють вам визначити, як дані повинні виглядати, які властивості вони мають, які є обов'язковими та унікальними.

В лістингу 3.19 наведена створена модель для книг. Дана схема містить title, author та destription всі вони мають тип даних String.

Лістинг 3.19 –Модель books.js

```

const mongoose = require('mongoose');

const bookSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true
  },
  author: {
    type: String,
    required: true
  },
  description: {
    type: String,
    required: true},
});
const Book = mongoose.model('Book', bookSchema);
module.exports = Book;

```

В лістингу 3.20 визначається схема для ролей. Роль має поле value типу String, яке є унікальним та має значення за замовчуванням "USER". Модель Role створюється на основі цієї схеми.

Лістинг 3.20 – Модель Role.js

```
const {Schema, model} = require('mongoose')
const Role = new Schema ({
  value: {type: String, unique: true, default: "USER"}
})
module.exports = model('Role', Role)
```

В лістингу 3.21 визначається схема для ролей. Роль має поле value типу String, яке є унікальним та має значення за замовчуванням "USER". Модель Role створюється на основі цієї схеми.

Лістинг 3.21 –Модель user.js

```
const {Schema, model} = require('mongoose')
const User = new Schema ({
  username: {type: String, unique:true, require: true},
  password: {type: String, require: true},
  roles: [{type: String, ref: 'Role'}]
})

module.exports = model('User', User)
```

3.2Реалізація дизайну вебсайту

3.2.1 Головна сторінка додатку

Дана сторінка вебсайту являє собою основну сторінку вебсайту, яку користувач баче при переході на наш вебсайт. На описана основна інформація про нього та основні елементи навігації

В лістингу 3.22 описуються основні налаштування нашої сторінки:

– <!DOCTYPE html> - Наша сторінка являє собою HTML5 документ;

- `<meta charset="UTF-8">` - вказує на кодування UTF-8 яка включає в себе великий спектр символів;
- `<meta name="viewport" content="width=device-width, initial-scale=1.0">` - дане налаштування створює основні параметри для відображення на різних пристроях.

Лістинг 3.22 – Модель Role.js

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Book List</title>
```

В наведеному лістингу 3.23 описується головні частини які мають на нашому додатку та опис книг які ми маємо.

Лістинг 3.23 – Описання нашого вебсайту

```
<h1> Ласкаво просимо в BookHaven!</h1>

<h2>Дослідіть світ слів разом із нами.</h2>

<div>
  <h3>Про нас</h3>
  <p>BookHaven - це цифрова бібліотека, де сторінки книжок
стають мостом між світами. Ми створили цей простір для тих, хто
цінує знання та мистецтво слова.</p>
</div>

<div>
  <h3>Наші колекції</h3>
  <ul>
    <li><strong>Різноманітність жанрів:</strong> Від
літературних шедеврів до останніх бестселерів - ми маємо щось для
кожного читача.</li>
    <li><strong>Освіта без кордонів:</strong> Збагатіть свій
розум з нашими освітніми ресурсами, включаючи підручники, статті та
курси.</li>
    <li><strong>Зручність читання:</strong> Читайте будь-де
і в будь-який час з нашим зручним онлайн-інтерфейсом.</li>
  </ul>
</div>

<div>
  <h3>Особливості</h3>
  <ul>
```

```

        <li><strong>Легкий пошук:</strong> Знайдіть цікаві для
вас твори за допомогою потужного пошуку і зручних фільтрів.</li>
        <li><strong>Ваші особисті колекції:</strong> Створюйте
власні бібліотеки, діліться своїми враженнями та отримуйте
персональні рекомендації.</li>
        <li><strong>Обговорюйте разом із нами:</strong>
Приєднуйтеся до обговорень, беріть участь в онлайн-клубах читання та
діліться своїми враженнями з іншими читачами.</li>
    </ul>
    <button id="buton1">Список книг</button>
    <button id="buton2">Додати</button>
    <button id="buton3">Задати питання</button>

```

Даний лістинг 3.24 доповнює попередній особливостями та кнопками навігації по основні сторінках. Кнопки посилення введуть до таких сторінок як Список книг, Додати, Задати питання.

Лістинг 3.24 – Описання переваг та кнопки переходу до основним компонентів

```

    <h3>Особливості</h3>
    <ul>
        <li><strong>Легкий пошук:</strong> Знайдіть цікаві
вас твори за допомогою потужного пошуку та зручних
фільтрів.</li>
        <li><strong>Ваші особисті колекції:</strong>
Створюйте власні бібліотеки, діліться своїми враженнями і
отримуйте персональні рекомендації.</li>
        <li><strong>Обговоріть разом з нами:</strong>
Приєднуйтеся до обговорень, приймайте участь в онлайн-клубах
чтень і діліться своїми враженнями з іншими читачами.</li>
    </ul>
    <button id="buton1">Список книг</button>
    <button id="buton2">Додати </button>
    <button id="buton3">Задати питання</button>

```

В лістинг 3.25 написаний скрипт для переходу до різних сторінок. За допомогою нього відбувається перехід до інших сторінок за допомогою кнопок в даній формі

Лістинг 3.25 – Описання переваг та кнопки переходу до основним компонентів

```

document.getElementById('buton1').onclick = function() {

```

```

window.location.href = 'http://localhost:5000/books';
}
document.getElementById('buton2').onclick = function() {
window.location.href = 'http://localhost:5000/add-book';
}
document.getElementById('buton3').onclick = function() {
window.location.href = 'http://localhost:5000/ask';
}

```

Лістинг 3.26 стилізує основні об'єкти коду наведеного вище. Таких як `Body`, `Header`, `section` – в даному блоці стилізується весь контент який знаходиться в даному блоці. Такий як шрифт тексту, задній фон, вирівнювання тексту та інше.

Лістинг 3.26 –Стилізація об'єктів

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background: rgb(224,140,177);
background: radial-gradient(circle, rgba(224,140,177,1) 0%,
    rgba(160,193,232,1) 100%);
}

header {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 1em;
}

section {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

```

В лістингу 3.27 наведена стилізація основних об'єктів на даній системі:

- H1 та H2 задається темний колір тексту для цих заголовків;
- UL та LI в цій частині забирається номер списку та встановлюється нульовий відступ. Також встановлюється відступ між елементами в даному списку;

- в блоці Footer вирівнює текст по центру, встановлює зовнішній відступ, внутрішній відступ, колір фону, та колір тексту для футера;
- Button задає основні стилі для кнопок які знаходяться на даній сторінці.

Лістинг 3.27 – Стилiзація об'єктiв

```
h2 {  
  color: #333;  
}  
  
div {  
  margin-bottom: 20px;  
}  
  
h3 {  
  color: #333;  
}  
  
ul {  
  list-style-type: none;  
  padding: 0;  
}  
  
li {  
  margin-bottom: 8px;  
}  
  
footer {  
  text-align: center;  
  margin-top: 20px;  
  padding-top: 20px;  
  background-color: #333;  
  color: #fff;  
}  
  
button {  
  background-color: #4CAF50;  
  color: white;  
  padding: 10px 20px;  
  border: none;  
  border-radius: 5px;  
  cursor: pointer;  
  margin: 5px;  
}  
  
button:hover {  
  background-color: #45a049;  
}
```

3.2.2 Форма реєстрації користувачів

У файлі `regist.hbs` рендериться форма для реєстрації користувачів. На даній сторінці знаходяться 2 форми, які заповнює користувач та кнопка, яка відправляє запит для виконання скрипту.

Лістинг 3.28 відповідає за відображення форми для введення даних для реєстрації а також знаходиться кнопка для запуску скрипту для відправки даних до обробляючого файлу.

Лістинг 3.28 – Форми для реєстрації та кнопка запуску скрипту

```
<title>Реєстрація</title>
<h2>Реєстрація</h2>
<form id="registration-form">
  <input type="text" name="username" placeholder="Логин" />
  <input type="password" name="password" placeholder="Пароль" />
  <button type="submit" class="btn">Зареєструватися</button>
</form>
```

Код наведений в лістингу 3.29 використовується на клієнтській стороні для обробки реєстраційного запиту і виводить сповіщення користувачеві.

Лістинг 3.29 – Скрипт для обробки даних

```
document.getElementById('registration-
form').addEventListener('submit', async (event) => {
  event.preventDefault(); // Предотвращаем стандартное действие
  отправки формы

  const username = document.querySelector('input
  [name="username"]').value;
  const password = document.querySelector('input
  [name="password"]').value;

  const data = {
    username: username,
    password: password
  };

  try {
    const response = await fetch('./auth/registration', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
```

```

        body: JSON.stringify(data)
    });

    if (response.ok) {
        // Успішно зареєстровані
        alert('Реєстрація виконана успішно ');
    } else {
        // Обробка помилок при реєстрації
        const responseData = await response.json();
        alert('Помилка під час реєстрації: ' +
responseData.message);
    }
    } catch (error) {
        console.error('Помилка при надсиланні даних ', error);
    }
})

```

В лістингу 3.30 стилізується задній фон цього файлу. В даній частині реалізується градієнт для заднього фону.

Лістинг 3.30 –Стилізація об'єктів

```

body {
    background: rgb(224,140,177);
background: radial-gradient(circle, rgba(224,140,177,1) 0%,
rgba(160,193,232,1) 100%);}

```

3.2.3Відображення детальної інформації про книги

У файлі bookDetails.hbs відображається основна інформацію про книги які знаходяться на нашому вебсайті. Інформацію для цього додаток бере із бази даних яку ми створимо.

В наданому лістингу 3.31 написаний шаблон в якому буде відображатися інформація детально інформацію. Для створення цього шаблону використовується шаблонізатор handlebars.

Лістинг 3.31 – Шаблон для детальної інформації про книги

```

<title>Book Details</title>
<div class="rectangle">
    {{#if message}}
    <p>{{message}}</p>
    {{else}}
    <h1>{{books.title}}</h1>

```

```

    <p>Author: {{books.author}}</p>
    <p>Description: {{books.description}}</p>
  {{/if}}
</div>

```

Лістинг 3.32 відповідає та створення градієнту для фону на даній сторінці, а також створення білого блока для контенту.

Лістинг 3.32 – Стилiзація об'єктiв

```

body {
  background: rgb(224,140,177);
  background: radial-gradient(circle, rgba(224,140,177,1) 0%,
    rgba(160,193,232,1) 100%);}
.rectangle {
  display: inline-block;
  background-color: #f0f0f0;
  border: 1px solid #ccc;
  box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.1);
  margin: 20px;
  padding: 20px;
  border-radius: 10px;
  width: 50%;
  height: 50vh;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);}

```

3.2.4 Реалізація форми для додавання книг до бази даних

Для того щоб книгу не потрібно було додавати до нашої бази даних через сайт mongodb було розроблено цю сторінку. За допомогою неї ми можете додати багато різних книжок. Для цього були створені форми для запису основної інформації та скрипт для додавання до баз даних.

В лістингу 3.33 створюються основні налаштування для цієї сторінки, які ми описували в лістингах вище. Також створюються форми для додавання основної інформації для книг, які будуть передані до скрипту який буде записувати їх в базу даних.

Лістинг 3.33 – Створення форм для додавання книг

```

<div class="rectangle">
  <div class="container">
    <h1>Add Book</h1>
    <form action="/add-book" method="POST">
      <label for="title">Title:</label>
      <input type="text" id="title" name="title" required>
      <br>
      <label for="author">Author:</label>
      <input type="text" id="author" name="author" required>
      <br>
      <label for="description">Description:</label>
      <textarea id="description" name="description"
required></textarea>
      <br>
      <button type="submit">Add Book</button>
    </form>
  </div>
</div>

```

В даному лістингу 3.34 записані основний CSS код для самого тіла та основних блоків цієї сторінки. Створюється градієнт для заднього фону, а також основний контент буде в білому блоці.

Лістинг 3.34 – Стилзація основних елементів

```

body {
  background: rgb(224,140,177);
  background: radial-gradient(circle, rgba(224,140,177,1)
0%, rgba(160,193,232,1) 100%);
}

rectangle {
  display: inline-block;
  background-color: #f0f0f0;
  border: 1px solid #ccc;
  box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.1);
  margin: 20px;
  padding: 20px;
  border-radius: 10px;
}

.container {
  text-align: center;
}

textarea {
  width: calc(100% - 22px);
  height: 150px;
  padding: 10px;
  box-sizing: border-box;
}

```

```

border: 1px solid #ccc;
border-radius: 5px;
resize: none;
font-family: 'Arial', sans-serif;
margin-bottom: 10px;
}

label {
display: block;
margin-bottom: 10px;
font-weight: bold;
}

```

Стили наведені в лістингу 3.35 визначають оформлення для елементів `<input>` (текстові поля для введення однорядкового тексту) та `<button>` (кнопки) у CSS.

Лістинг 3.35 – Стилiзація input button

```

input {
width: calc(100% - 22px);
padding: 10px;
box-sizing: border-box;
border: 1px solid #ccc;
border-radius: 5px;
margin-bottom: 10px;
}

button {
padding: 10px;
background-color: #4caf50;
color: white;
border: none;
border-radius: 5px;
cursor: pointer;
}

button:hover {
background-color: #45a049;
}

```

3.2.5 Форма для запитання у ChatGPT

Ця форма створена для вебінтерфейсу, який дозволяє користувачеві вводити запитання, а потім відправляти ці запитання до серверної частини (з використанням `/askgpt`) для обробки запитань за допомогою OpenAI GPT-3.

В лістингу 3.36 наведена основна HTML-розмітка для нашої сторінки. Заголовок сторінки та форма, яку буде заповнювати користувач знаходяться всередині блока з класом "rectangle". Форма використовує метод post та вказує на /askgpt коли форма відправляється.

Лістинг 3.36 – HTML розмітка

```
<title>GPT-3 Chat</title>
<div class="rectangle">
<h1>GPT-3 Chat</h1>
<form action="/askgpt" method="post">
  <label for="question">Ваше питання:</label>
  <input type="text" id="question" name="question"
required>
  <button type="submit">Надіслати питання</button>
</form>
</div>
```

В даному лістингу 3.37 визначаються стилі для <body> та блока з класом "rectangle". Задається фон для <body> та стилі для блока форми. Рамка, тінь та округлення кутів надають більш привабливий вигляд.

Лістинг 3.37 – Стилзація HTML розмітки

```
body {
  background: rgb(224,140,177);
background: radial-gradient(circle, rgba(224,140,177,1) 0%,
rgba(160,193,232,1) 100%);
}

.rectangle {
  width: 600px;
  height: 300px;
  background-color: #f0f0f0;
  border: 1px solid #ccc;
  box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.1); /* Тінь
блока */
  margin: 20px;
  padding: 20px;
  border-radius: 10px; /*Радіус скруглення кутів блоку */}
```

3.2.6 Перелік книг, які знаходяться в базі даних

На даній сторінці рендериться всі книги які ми додали до нашого вебсайту за допомогою сторінки додавання книг. Всі книги знаходяться в нашій базі даних.

Сторінка (лістинг 3.38) містить заголовок "Список книг" та умовний блок `{{#if books}}`, який перевіряє наявність книг у базі даних. Якщо книги існують, вони виводяться у вигляді списку з деталями та посиланням на деталі кожної книги. Якщо книг немає, виводиться повідомлення "No books available". Додається посилання "Add a Book", яке веде на сторінку додавання нової книги.

Лістинг 3.38 – Основна HTML розмітка для сторінки

```
<h1>Список книг</h1>

{{#if books}}
  <ul>
    {{#each books}}
      <p>{{title}}, {{author}}, {{year}} <a
href="/books/{{_id}}">Details</a></p>
    {{/each}}
  </ul>
{{else}}
  <p>No books available.</p>
{{/if}}

<a href="/add-book">Add a Book</a>
```

Лістинг 3.39 визначаються стилі для `<body>` та класу `.rectangle`. Встановлено фон для `<body>` з використанням градієнту. Клас `.rectangle` використовується для стилізації блоків, які, використовуються для представлення окремих книг у списку.

Лістинг 3.39 – Стилiзація цієї сторінки

```
body {
  background: rgb(224,140,177);
  background: radial-gradient(circle, rgba(224,140,177,1) 0%,
    rgba(160,193,232,1) 100%);
}

.rectangle {
  width: 300px; /* Ширина блока */
  height: 150px; /* Висота блока */
  background-color: #f0f0f0; /* Колір фону блока */
  border: 1px solid #ccc; /* Границя блока */
  box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.1); /* Тінь
блока */
  margin: 20px; /* Відступи блока */
  padding: 20px; /* Внутрішній відступ блока */
  border-radius: 10px; /* Радіус скруглення кутів блока */
}
```

```

    }

body {
    background: rgb(224,140,177);
    background: radial-gradient(circle, rgba(224,140,177,1) 0%,
    rgba(160,193,232,1) 100%);
}

```

3.2.7 Форма авторизації

В даному файлі рендериться форма для авторизації користувачів. На даній сторінці знаходяться 2 форми, які заповнює користувач та кнопка яка відправляє запит для виконання скрипту.

Даний лістинг 3.40 є HTML-структура для форми авторизації з полем для введення імені користувача, полем для введення паролю та кнопкою для відправлення. Додатково є два елементи div з ідентифікаторами "messages" і "loading", призначені для відображення повідомлень та завантаження.

Лістинг 3.40 – Форми авторизації

```

<div class="container">
  <div class="rectangle">
    <h2>Авторизація</h2>
    <form id="authorization-form">
      <input type="text" title="username"
placeholder="username" />
      <input type="password" title="Логин" placeholder="Пароль"
/>
      <button type="submit" class="btn">Авторизуватися</button>
    </form>
    <!-- Add a div for displaying messages and a loading
spinner -->
    <div id="messages"></div>
    <div id="loading">Loading...</div>
  </div>
</div>

```

Цей лістинг 3.41 встановлює фон для сторінки, стилі для прямокутника (відображається як форма авторизації), а також центрує текст всередині контейнера. Спеціальний блок #loading встановлює стилі для спінера завантаження із властивістю display: none, що приховує його на початку.

Лістинг 3.41 – Стилiзація сторiнки

```
body {
    background: rgb(224,140,177);
    background: radial-gradient(circle, rgba(224,140,177,1)
0%, rgba(160,193,232,1) 100%);
}

.rectangle {
    display: inline-block;
    background-color: #f0f0f0;
    border: 1px solid #ccc;
    box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.1);
    margin: 20px;
    padding: 20px;
    border-radius: 10px;
}

.container {
    text-align: center;
}

/* Add styles for the loading spinner */
#loading {
    display: none;
    margin-top: 10px;
}
```

В даному лістингу 3.42 виконується скрипт для обробки інформації з форми сторінки. Цей код відсилає оброблену інформацію на сервер, якщо під час виконання відбудеться помилка її виведе користувачеві.

Лістинг 3.42 – Скрипт для обробки інформації

```
const form = document.getElementById('authorization-form');

const messagesDiv = document.getElementById('messages');

const loadingDiv = document.getElementById('loading');

form.addEventListener('submit', async (event) => {
    event.preventDefault();

    const username = document.querySelector('input
[title="username"]').value;

    const password = document.querySelector('input
[title="Логин"]').value;

    const data = {
        username: username,
```

```

        password: password
    };

    // Show loading spinner
    loadingDiv.style.display = 'block';
    messagesDiv.innerHTML = ''; // Clear previous messages

    try {
        // Simulate an asynchronous request
        await new Promise(resolve => setTimeout(resolve,
1000));

        // In a real scenario, replace the following block with
the actual fetch request
        const response = { ok: true }; // Replace with your
server response

        if (response.ok) {
            alert('Авторизація успішно виконана');
        } else {
            messagesDiv.innerHTML = 'Помилка при авторизації';
        }
    }
    catch (error) {
        console.error('Помилка при відправці даних', error);
    } finally {
        // Hide loading spinner
        loadingDiv.style.display = 'none';
    }
    });

```

3.3 Створення бази даних MongoDB

Mongodb - це документно-орієнтована система управління базами даних. Для зберігання даних використовується JSON формат, що є дуже зручним для розробників. Особливо цей СУБД відрізняється високою доступністю, масштабістю і безпекою. Серед головних особливостей можна відзначити, що вона швидко розгортається в облаці. Основний плюс цієї системи що не вимагає опису схеми таблиці, як у революційних базах даних. Дані зберігаються у вигляді колекцій і документів.

MongoDB Atlas

- ✓ **Work with your data as code**
Documents in MongoDB map directly to objects in your programming language. Modify your schema as your apps grow over time.
- ✓ **Focus on building, not managing**
Let MongoDB Atlas take care of the infrastructure operations you need for performance at scale, from always-on security to point-in-time recovery.
- ✓ **Simplify your data dependencies**
Leverage application data for full-text search, real-time analytics, rich visualizations and more with a single API and minimal data movement.

Sign up

See what Atlas is capable of for free

Sign up with Google

First Name*

Last Name*

Company

Email*

Password*

Рисунок 3.1 - Реєстрація

Для початку роботи з MongoDB потрібно зареєструватися на офіційному вебсайті. На рисунку 3.1 зображена сторінка реєстрації. Для цього потрібно заповнити всі поля на даній сторінці.

Після реєстрації ви опинитесь на стартовій сторінці (рисунку 3.2), на якій будуть основна навігація по нашому проєкту. Для створення нової бази даних потрібно на сторінці натиснути кнопку «New project».

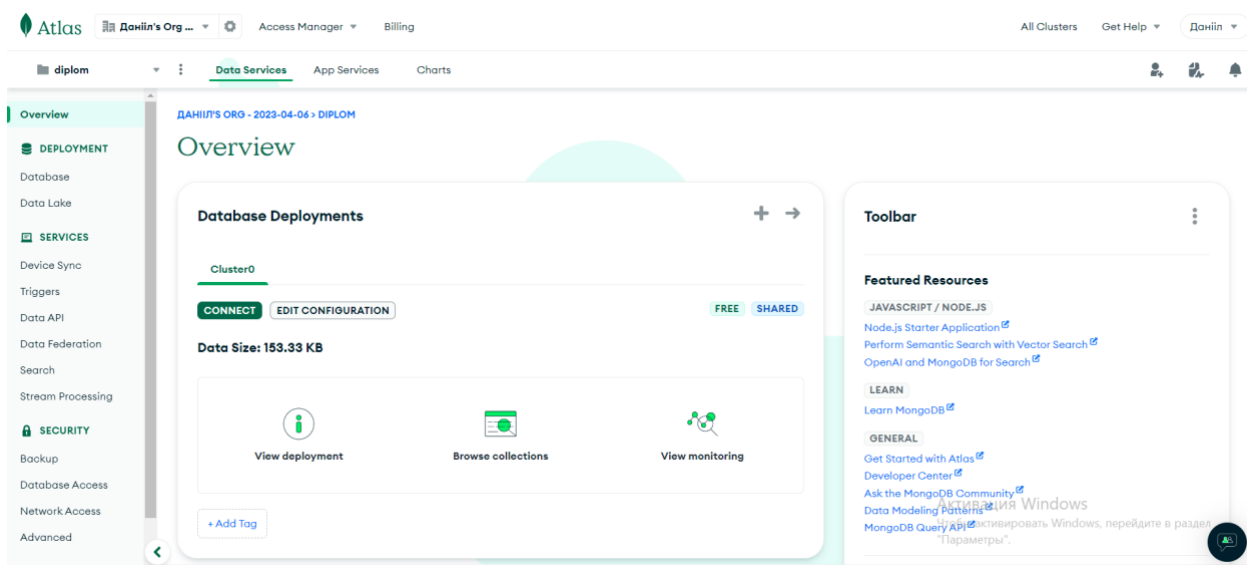


Рисунок 3.2- Стартова сторінка.

На рисунку 3.1 відбувається створення нового проєкту для цього потрібно ввести назву нашої бази даних. Під цим ім'ям ми будемо відсилати запити з нашого додатку.

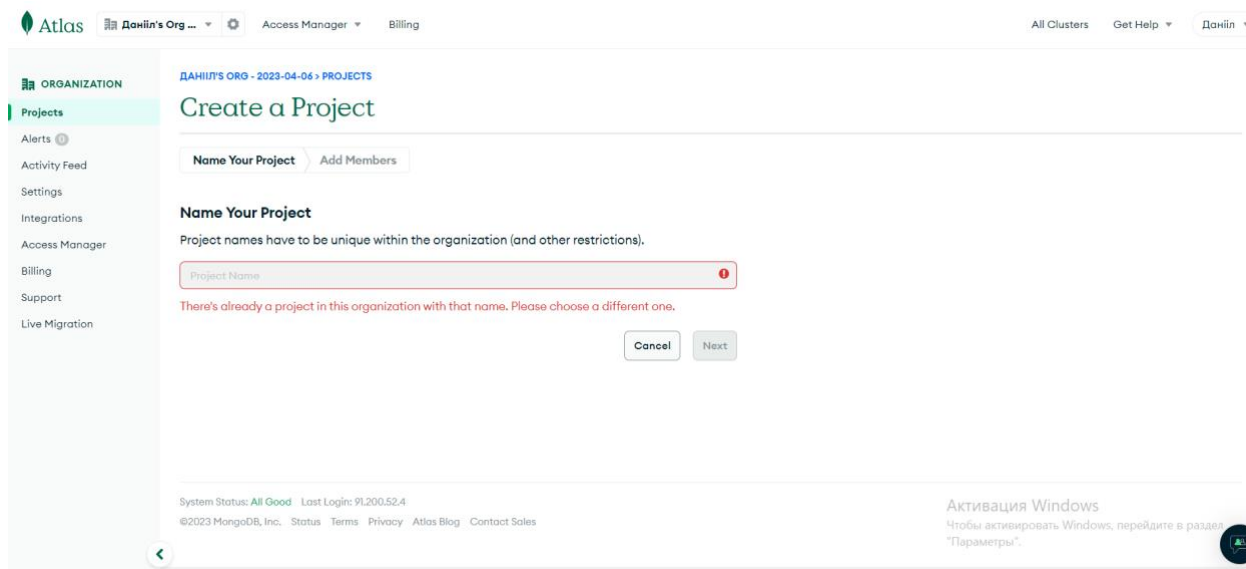


Рисунок 3.3 - Створення назви проєкту

Після того, як ми введемо ім'я нашого проєкту налаштуємо основні параметри для нашої бази даних. На рисунку 3.1 зображені параметри платного або безплатного проєкту, вибір хостингу на якому будуть розміщені наша база, також вибір міста, в якому він буде розташований.

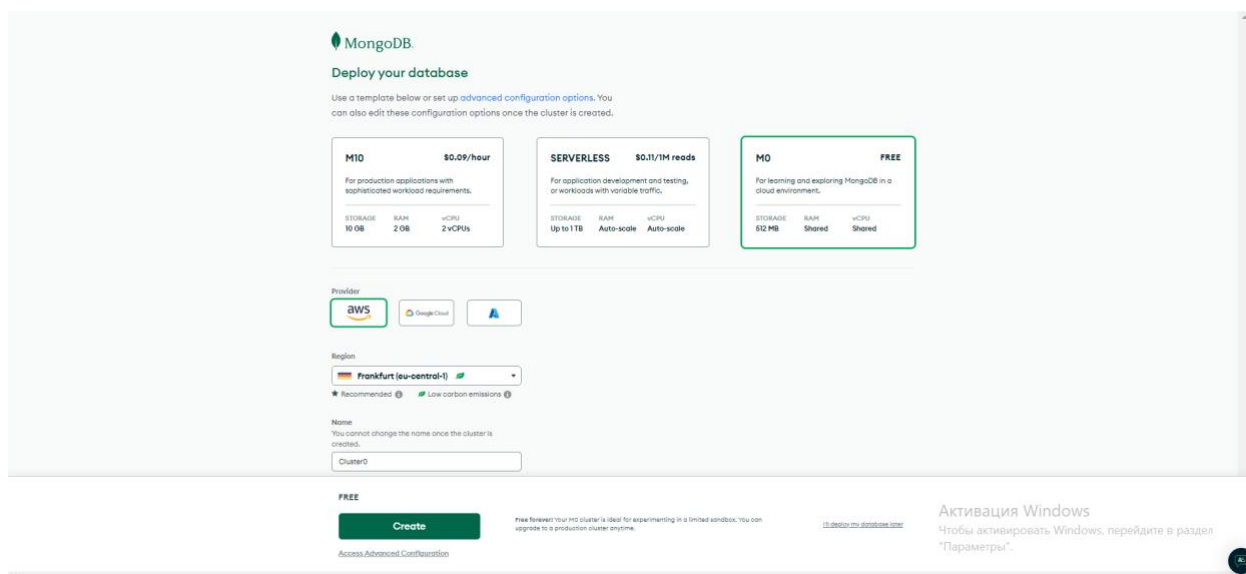


Рисунок 3.4 - Основні налаштування

Коли ми виконаємо основні настройки хостингу ми повинні додати ір-адреси які будуть взаємодіяти з нашою базою(рисунок 3.1). Для цього ми можемо використати додати ір-адресу, з якої ми зараз користуємося вебсайтом. Але це можна використати якщо ваш сервер знаходиться на цій адресі. Для більш точності ми можемо самостійно ввести адресу, або використати функцію яка буде дозволяти використовувати всі адреси.

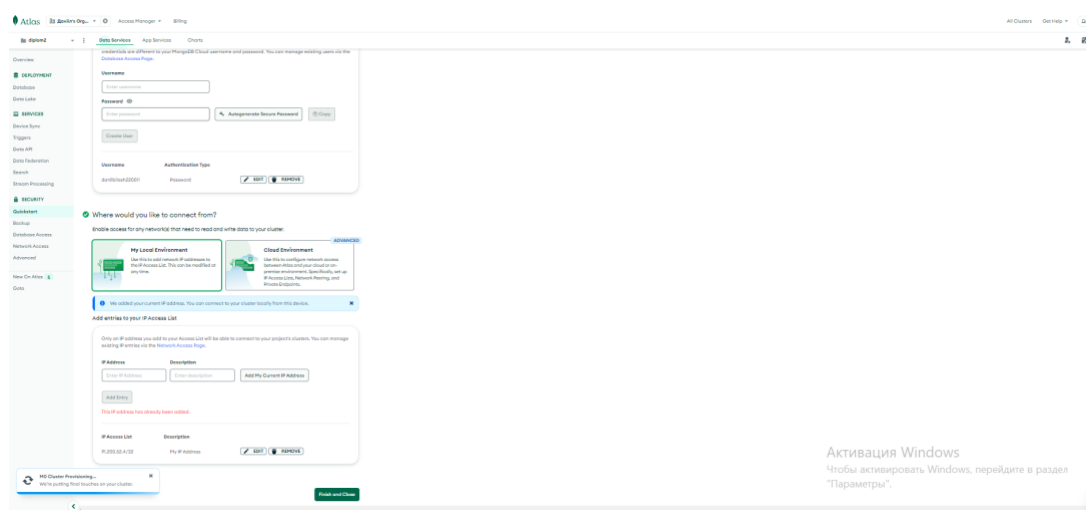


Рисунок 3.5 - Налаштування ір-адрес

Після всіх цих маніпуляцій в нас буде створена база даних, в якій будуть знаходитися наші колекції. В даних колекція будуть зберігатися наші дані. На даній сторінці ми зможемо переглянути всі дані, які знаходяться тут, а також ми зможемо редагувати вміст наших колекцій.

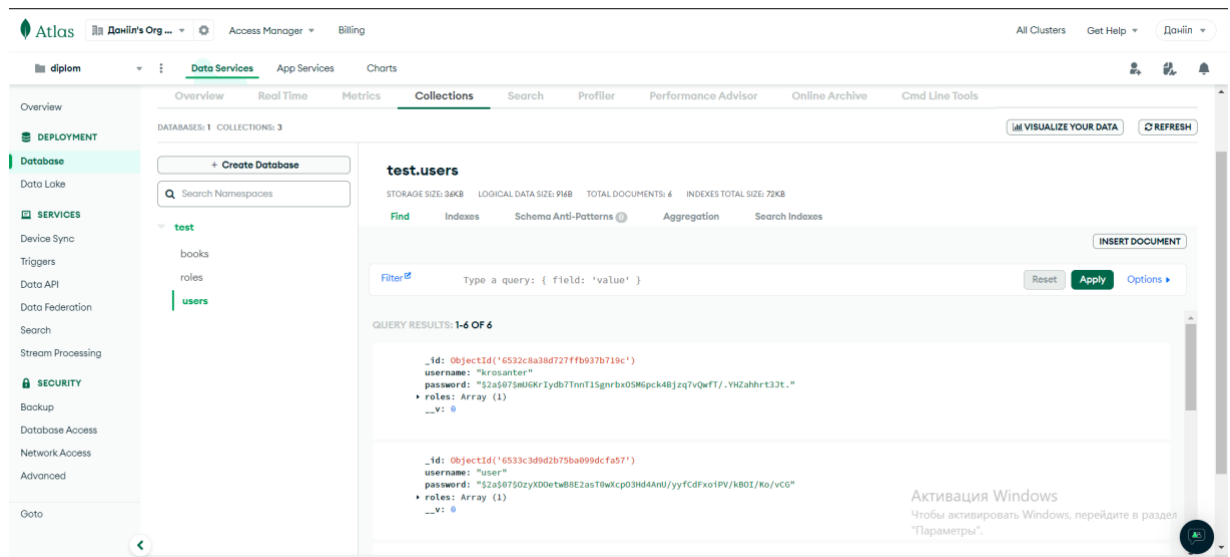


Рисунок 3.6 - Колекції проєкту

4 РЕЗУЛЬТАТИ РОБОТИ

У результаті виконання дипломного проєкту було реалізовано основний функціонал сайту: головну сторінку (рисунок 4.1) сайту, механізм авторизації та реєстрації користувачів, додавання книжок до бази даних, перегляд інформації про книги, і було розроблено сторінку, на якій користувачі зможуть задати питання до інтелектуальному чатботу (рисунок 4.2).

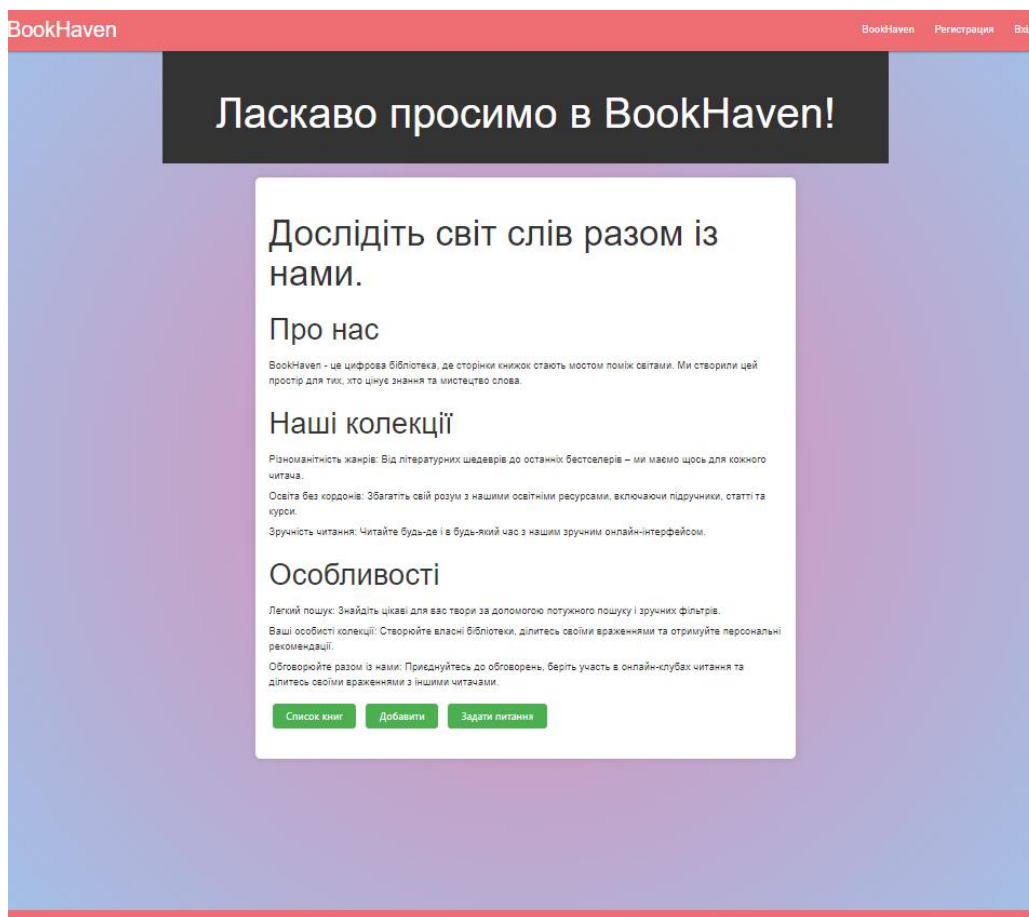


Рисунок 4.1 – Головна сторінка

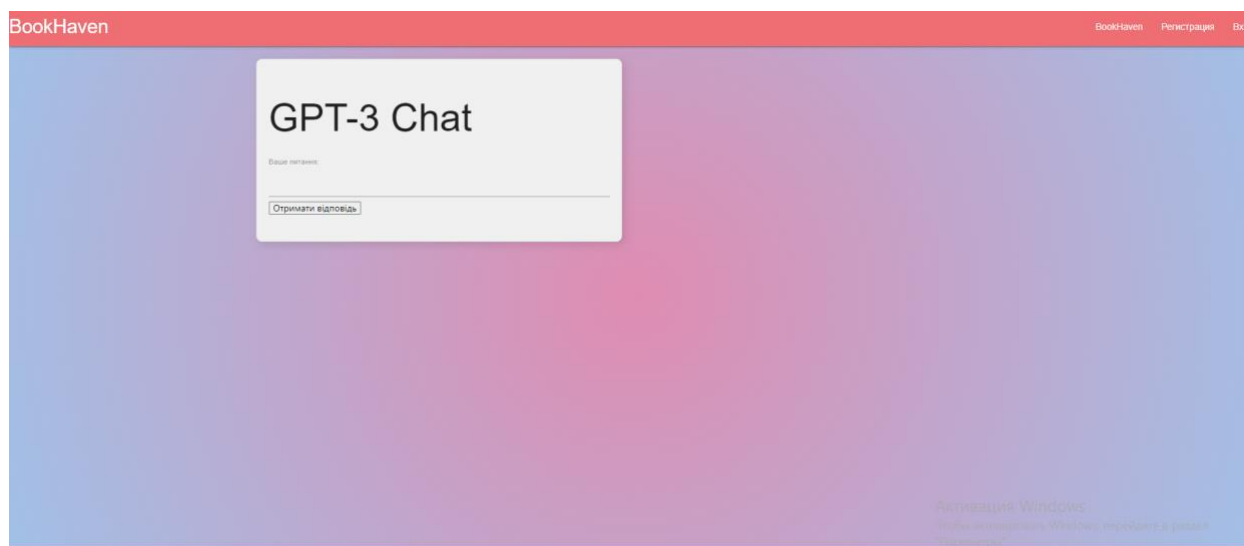


Рисунок 4.2 – Сторінка для запитання

Також в даному проєкті було підвищено надійність системи за допомогою використання сучасних технологій для розробки вебдодатків.

На рисунок 4.3 відображається форма для реєстрації користувачів. На цій сторінці розташовані дві форми, які користувач може заповнити, а також кнопка, призначена для відправлення запиту на виконання скрипту.

Під час реєстрації відбувається перевірка валідації:

- ім'я користувача не може бути порожнім;
- пароль має бути більше 4 і менше 10 символів.

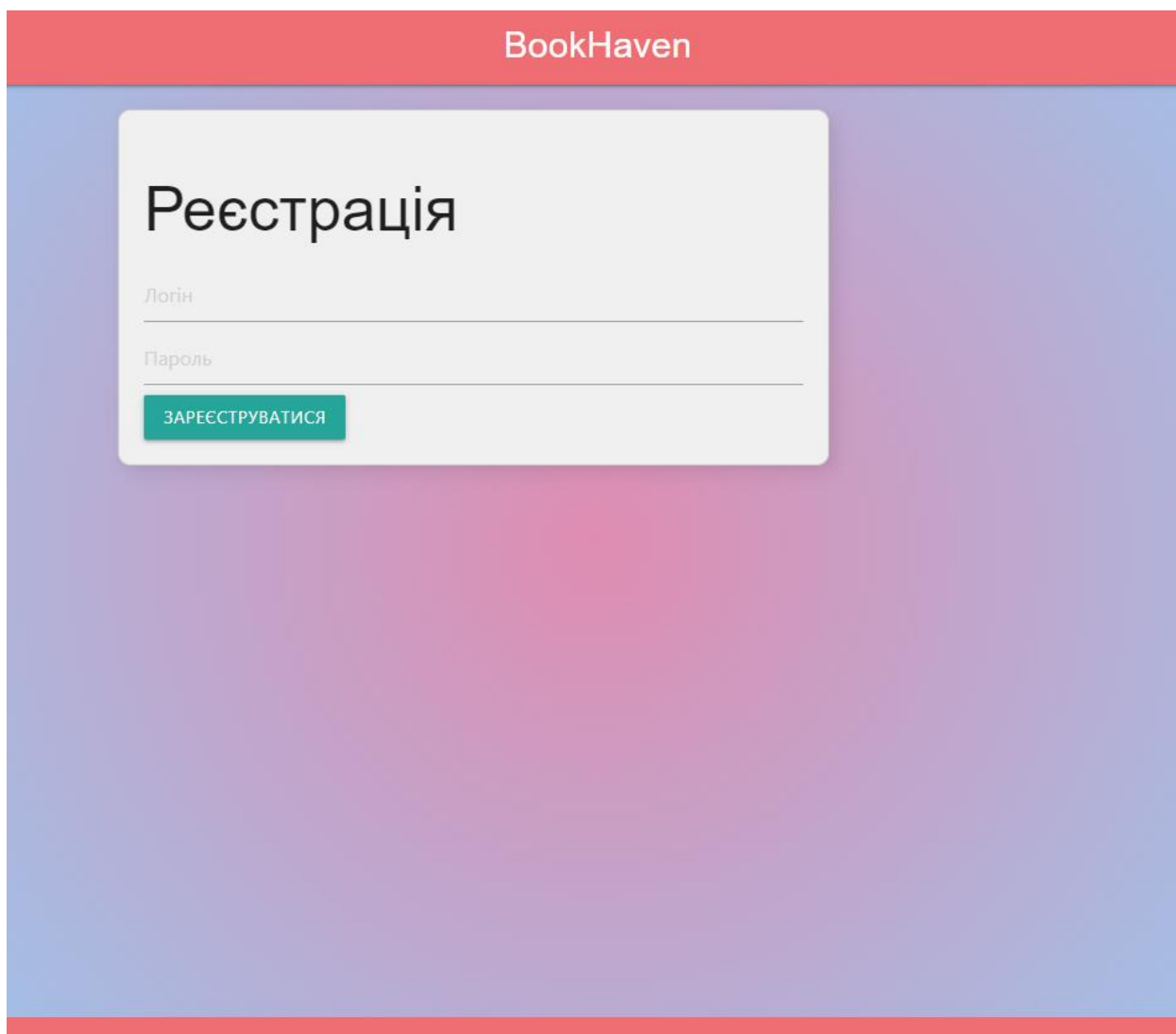


Рисунок 4.3 – Сторінка реєстрації

Після успішної реєстрації, користувач має можливість авторизуватися на окремій сторінці (рисунок 4.4) за допомогою введення свого логіну та паролю.

Також в даній роботі реалізовано авторизація під різні ролі на вебсайті. При реєстрації автоматично створюється роль USERS, а роль ADMIN вже записана в базу даних при створенні проєкту.

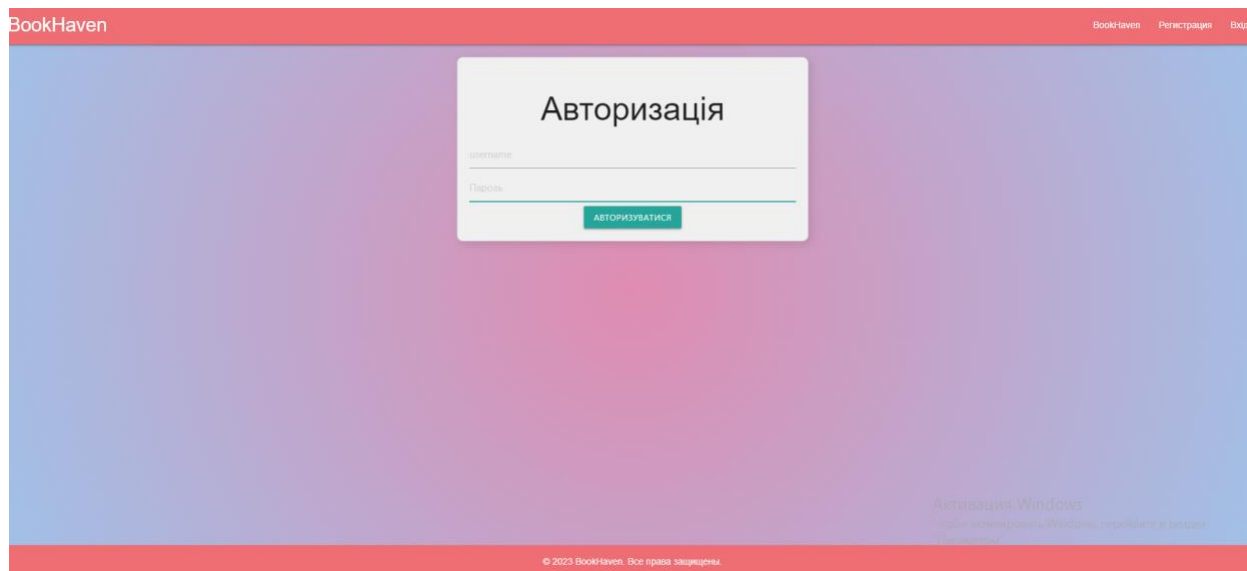


Рисунок 4.4 – Авторизації користувача

Під час авторизації розроблений блок повідомлень про помилки:

- якщо користувач буде вводити логін якого немає в базі даних виведеться помилка “Користувач не знайдений”;
- коли логін введено правильно, а пароль ні помилка виведеться помилка “Неправильний пароль”.

Після того як користувач правильно заповнив поля буде повідомлення про успішну авторизацію (рисунок 4.5).

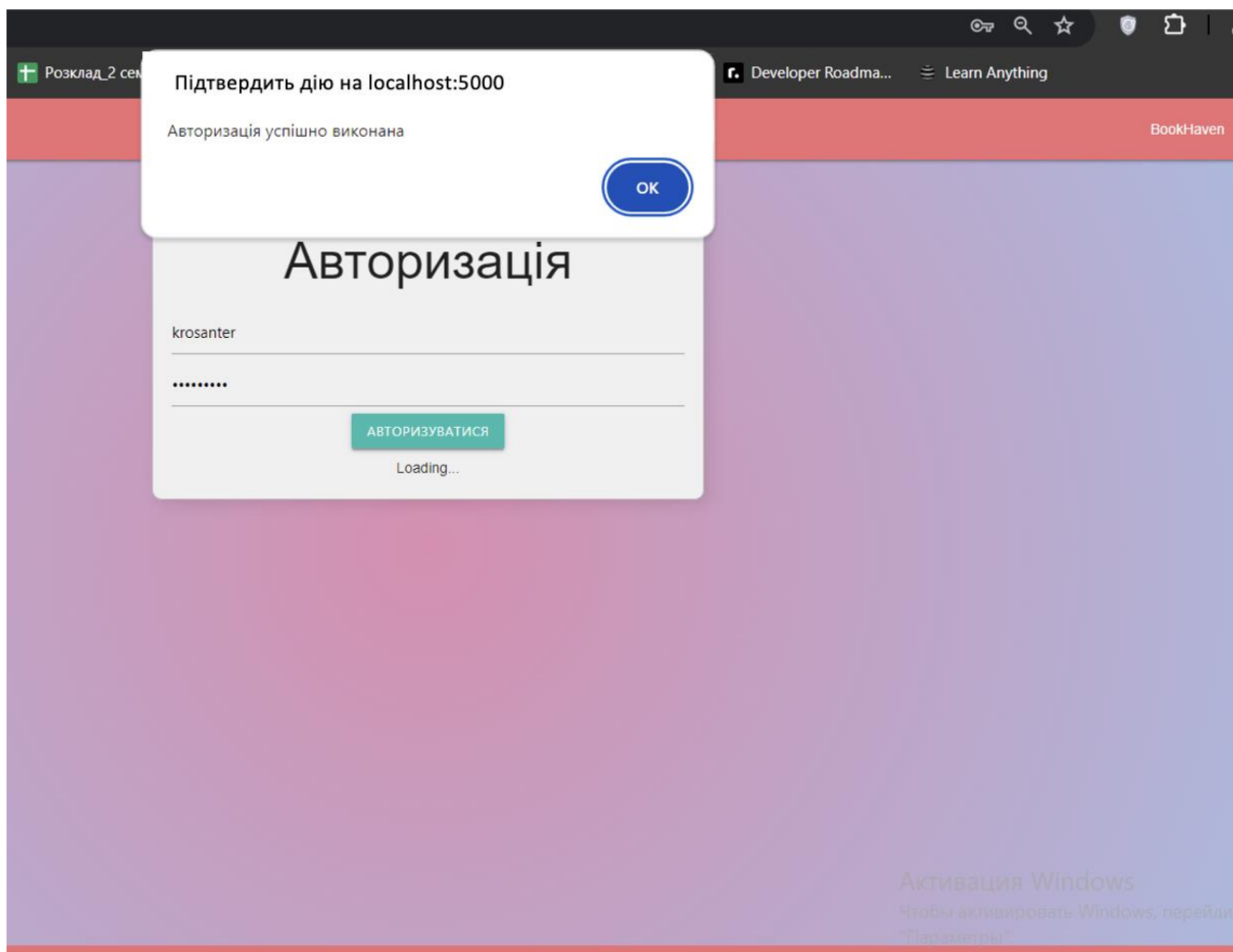


Рисунок 4.5 – Авторизації користувача

В даному проєкті також було підвищено надійність системи за допомогою хешування даних. За допомогою використання сучасної технології `bcrypt.js` яка забезпечила безпеку даних від кіберзлочинців, які зберігаються в базі даних.

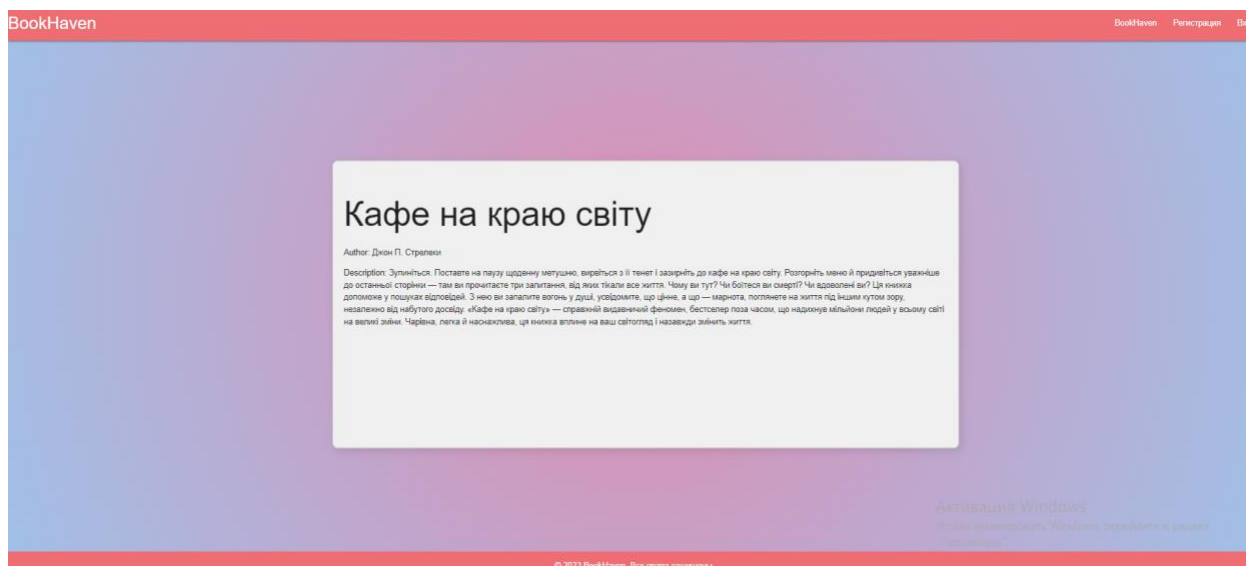


Рисунок 4.6 – Детальна інформація про книгу

На рисунок 4.6 зображена сторінка нашого вебсайту, на якій відображено детальна інформація про книги. Детальна інформація про книги знаходиться в базі даних. Запит до бази даних реалізований на серверній частині додатку.

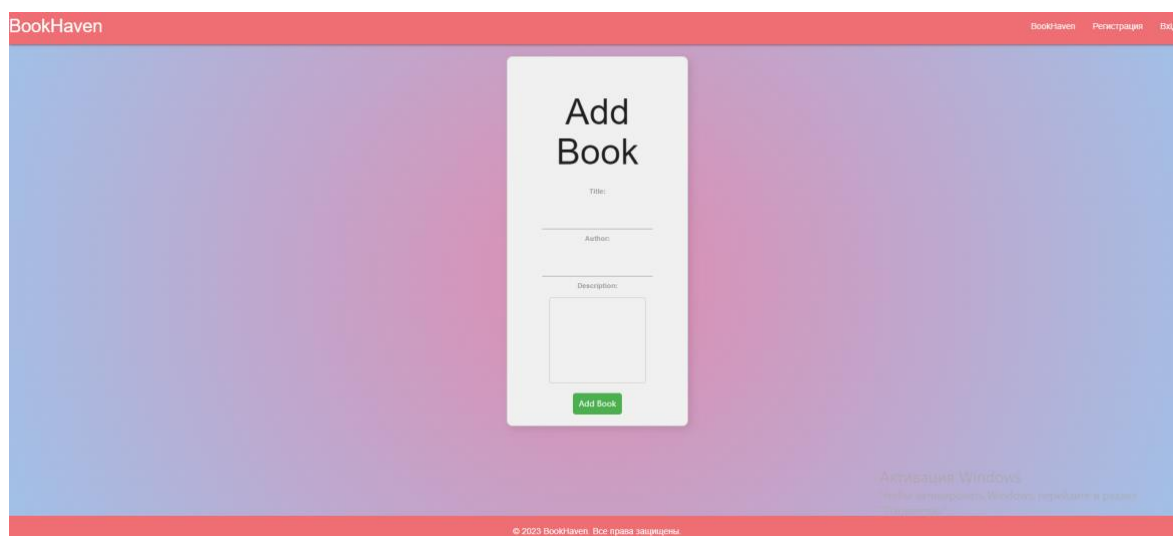


Рисунок 4.7 – Додавання книги

Для того щоб доповнювати асортимент книг, було розроблена сторінка додавання книг до бази даних (рисунок 4.7). Заповнюючи форми користувачі зможуть доповнювати їх.

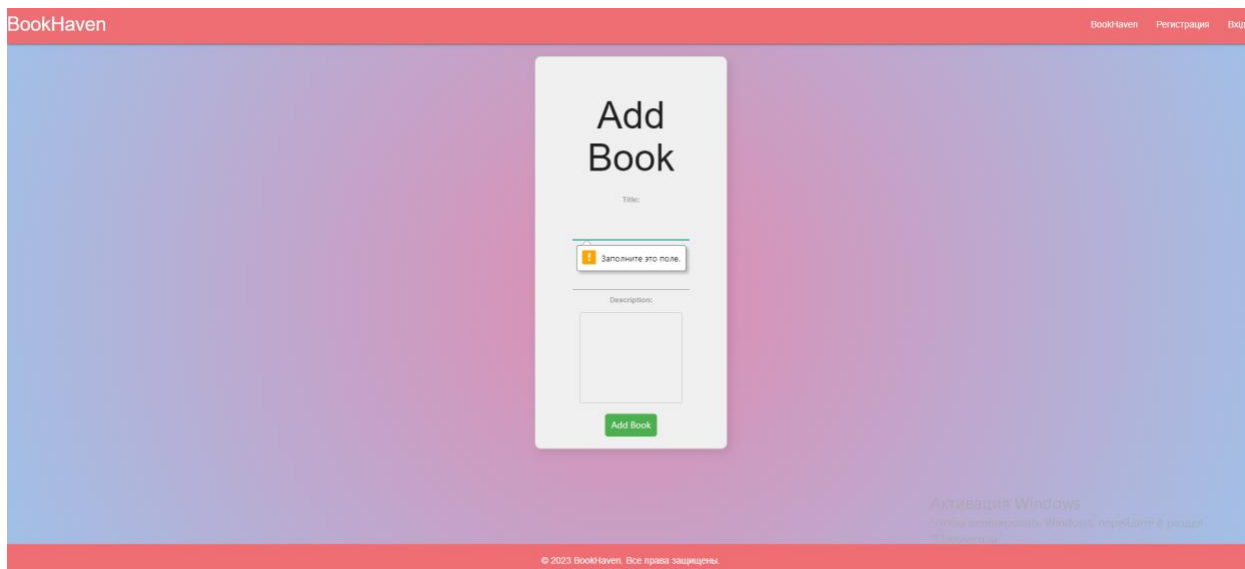


Рисунок 4.8 – Додавання книги

Також при додавання книги до бази даних потрібно не забувати додавати до бази даних заповнювати всі поля форми, інакше буде помилка (рисунок 4.8).

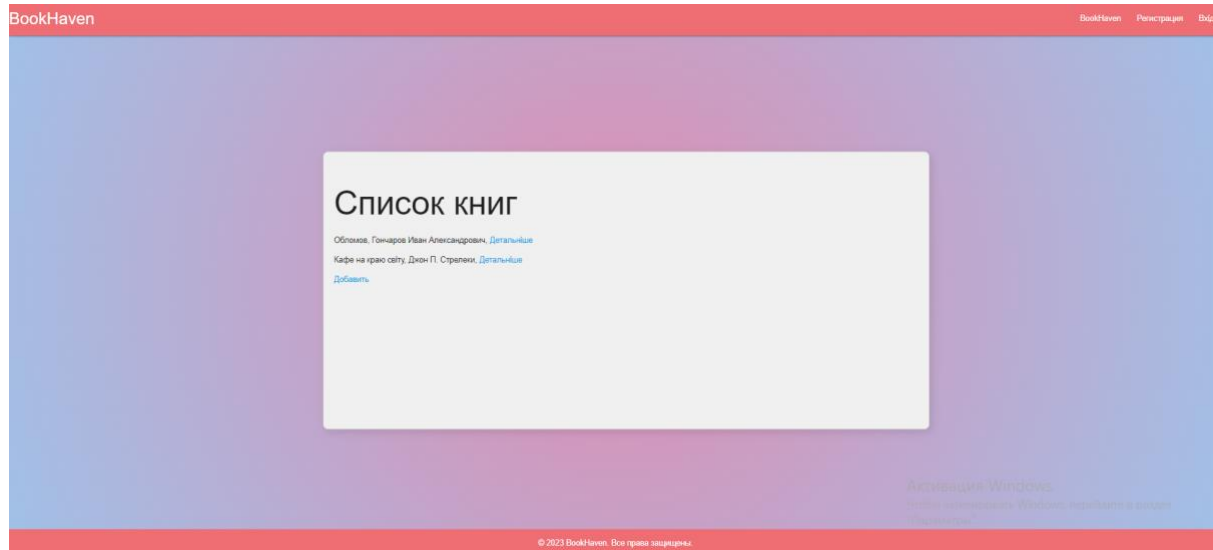


Рисунок 4.9 – Список книг на нашому вебсайті

Для ознайомлення з асортиментом книг було реалізовано сторінку зі списком книг (рисунок 4.9). Дані про книжки знаходяться в базі даних. Також напроти кожної книги можна перейти до сторінки з детальною інформацією про книгу.

ВИСНОВКИ

У даній дипломній роботі досліджено та впроваджено різні методи для досягнення більш надійних вебсайтів. Ця робота включає в себе аналіз та вдосконалення різних аспектів веб-розробки. Перш за все, вивчалася проблема надійності, зокрема, чинники, що призводять до відмов та збоїв. Також включає в себе аналіз інфраструктури серверів, використання резервних копій даних та заходів безпеки для запобігання атакам.

Враховуючи постійний розвиток сфери розробки потрібно вивчати нові методи, бо з кожним роком все більше і більше методів стають не актуальними бо їх витісняють більш сучасні та досконаліші в багатьох аспектах.

Використання штучного інтелекту є дуже важливим етапом для збільшення автоматизації та саморегулювання вебсайту. Це призвело до зменшення часу обробки запитів користувачів та отримання допомоги на вебсайті цілодобово.

Однією з ключових складових цього підходу є використання API для отримання текстових відповідей від штучного інтелекту. ChatGPT, як один із представників таких систем, здатен генерувати людино-подібний текст, розуміти запитання користувачів та надавати інформативні відповіді.

Після отримання відповідей ви використали їх обробку для створення більш зручного та знайомого для користувача вигляду. Це може включати в себе перетворення текстових відповідей у більш структурований формат, додавання графічних елементів або навіть використання додаткових мовних обробок для покращення зрозумілості та логічності відповідей.

Захист системи вимагає постійного моніторингу нових методів та адаптації до сучасних реалій і це реалізовано у системі. У даній роботі вдосконалено систему реєстрації та аутентифікації, яка грає критичну роль у забезпеченні високого рівня захисту інформаційних ресурсів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kocoń, Jan, et al. "ChatGPT: Jack of all trades, master of none." *Information Fusion* (2023): 101861.
2. George, A. Shaji, and AS Hovan George. "A review of ChatGPT AI's impact on several business sectors." *Partners Universal International Innovation Journal* 1.1 (2023): 9-23.
3. AlAfnan, Mohammad Awad, et al. "Chatgpt as an educational tool: Opportunities, challenges, and recommendations for communication, business writing, and composition courses." *Journal of Artificial Intelligence and Technology* 3.2 (2023): 60-68.
4. JavaScript. Повне керівництво, 7-е видання. Девід Фленаган. 2021. 722 с.
5. Zammetti, Frank. *Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker*. Apress, 2020.
6. Herron, David. *Node.js Web Development: Server-side web development made easy with Node 14 using practical examples*. Packt Publishing Ltd, 2020.
7. B Goswami, Pronnoy, et al. "Investigating the reproducibility of npm packages." 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2020.
8. Tabarés, Raúl. "HTML5 and the evolution of HTML; tracing the origins of digital platforms." *Technology in Society* 65 (2021): 101529.
9. Goldstein, Alexis, Louis Lazaris, and Estelle Weyl. *HTML5 & CSS3 For The Real World: Powerful HTML5 and CSS3 Techniques You Can Use Today!*. SitePoint Pty Ltd, 2015.
10. Rebah, Hassen Ben, Hamed Boukthir, and Antoine Chedebois. *Website Design and Development with HTML5 and CSS3*. John Wiley & Sons, 2022.
11. Білаш Д.І., Аналіз взаємодії між frontend та backend при розробці вебдодатків / Д.І. Білаш, М.Ю. Тягунова // *Технологія-2023 : міжнар. наук.-техн. конф.*, 26 травня 2023 : тези доп. – м. Київ,. – С. 57-59.

- 12.Chen, Beidi, et al. "Mongoose: A learnable lsh framework for efficient neural network training." International Conference on Learning Representations. 2020.
- 13.Mardan, Azat, and Azat Mardan. "Using Express. js to create Node. js web apps." Practical Node. js: Building Real-World Scalable Web Apps (2018): 51-87.
- 14.Manricks, Gabriel. Instant Handlebars. js. Packt Publishing Ltd, 2013.
- 15.Rahmatulloh, A., R. Gunawan, and F. M. S. Nursuwars. "Performance comparison of signed algorithms on JSON Web Token." IOP Conference Series: Materials Science and Engineering. Vol. 550. No. 1. IOP Publishing, 2019.
- 16.Baral, Purna. "Role-based User Access Control in MERN Stack applications." (2020).
- 17.Redmond Lawlor, William. The Ace Manager: Technical Report. Diss. Dublin, National College of Ireland, 2022.