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

Інститут електроенергетики
(інститут)
факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня бакалавра

(бакалавра, спеціаліста, магістра)

Студента Гайворонського Владислава Юрійовича
(ПІБ)

академічної групи 126-20-1
(шифр)

спеціальності 126 «Інформаційні системи та технології»
(код і назва спеціальності)

за освітньо-професійною програмою «Інформаційні системи та технології»
(офіційна назва)

на тему Розробка інтелектуального помічника з використанням хмарних сервісів
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікавційної роботи	проф. Коротенко Г.М.			
розділів:				

Рецензент	доц. Ширін А.Л.			

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри

інформаційних технологій
та комп'ютерної інженерії
(повна назва)

_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)

«_____»_____2024 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня _____бакалавр_____
(бакалавра, спеціаліста, магістра

студенту _____Гайворонському В.Ю._____ **академічної групи** ____126-20-1____
(прізвище та ініціали) (шифр)

спеціальності _____126 « Інформаційні системи та технології »_____

за освітньою-професійною програмою _____

_____«Інформаційні системи та технології»_____

на тему ___Розробка інтелектуального помічника з використанням хмарних сервісів___

_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 23.05.2024 р. № 469-с

Розділ	Зміст	Термін виконання
Розділ 1	Ознайомлення з матеріалами сайтів, літературою та виконання пошуку технологій і формування відповідних рішень	01.02.2024 – 30.03.2024
Розділ 2	Реалізація проектних рішень	01.04.2024 – 20.06.2024

Завдання видано _____ _____Коротенко Г.М._____
(підпис керівника) (прізвище, ініціали)

Дата видачі _____01.02.2024 р._____

Дата подання до екзаменаційної комісії ____02.07.2024 р.____

Прийнято до виконання _____ _____Гайворонський В.Ю._____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 66 сторінок, 20 рисунків, 5 таблиць, 3 додатка, 26 джерел.

Об’єкт розробки: застосунок інтелектуального помічника з використанням хмарних сервісів.

Мета кваліфікаційної роботи: розробка програмного забезпечення інтелектуального помічника.

У першому розділі звертається увага на актуальність проблеми, проводиться аналіз існуючих аналогів віртуальних помічників для вебсайтів, і визначаються мета та задачі кваліфікаційної роботи.

Другий розділ присвячений проектуванню інтелектуального помічника, де наведено опис його архітектури, алгоритму роботи, та функціональності. Крім того, розглядаються технології, які будуть використовуватись для реалізації проекту.

У наступному розділі детально описується реалізація застосунку. Виконується розробка самого інтелектуального помічника з використанням інструментів фреймворків Flask, мови програмування Python, та моделі штучного інтелекту Gemini AI. Також описується порядок взаємодії з графічним інтерфейсом користувача.

Результатом проведеної роботи є створений застосунок інтелектуального помічника для вебсайту книжкового інтернет-магазину. Цей застосунок реалізує функції віртуального асистента, який консультує відвідувачів вебсайту, та супроводжує процеси реєстрації та оформлення замовлення.

Ключові слова: ІНТЕЛЕКТУАЛЬНИЙ ПОМІЧНИК, ЧАТ, КОНСУЛЬТАНТ, ЗАСТОСУНОК, ІНТЕРНЕТ-МАГАЗИН, ВЕБСАЙТ, FLASK, GEMINI AI, PYTHON.

ABSTRACT

Explanatory note: 66 pages, 20 fig., 5 tabs, 3 appendices, 26 sources.

The object of development: intelligent assistant application using cloud services.

The purpose of the qualification work: development of intelligent assistant software.

The first section draws attention to the relevance of the problem, analyzes existing analogs analogues of virtual assistants for websites, and defines the purpose and tasks of the qualification work.

The second section is devoted to the design of the intelligent assistant, which describes its architecture, work algorithm, and functionality. In addition, the technologies that will be used for the implementation of the project are considered.

The next section describes the implementation of the application in detail. The development of the intelligent assistant using the tools of the Flask framework, the Dart programming language and the Gemini AI model. The order of interaction with the graphical user interface is also described.

The result of the work is the creation of an intelligent assistant application for the website of an online bookstore. This application implements the functions of a virtual assistant that advises website visitors and accompanies the registration and ordering processes.

Keywords: INTELLIGENT ASSISTANT, CHAT, CONSULTANT, APPLICATION, ONLINE STORE, WEBSITE, FLASK, GEMINI AI, PYTHON.

ЗМІСТ

ВСТУП

На сьогоднішній день багато вебсайтів пропонують певний рівень онлайн-обслуговування клієнтів. Іноді це просто онлайн-чат зі справжнім представником служби підтримки клієнтів, але все більше операторів вебсайтів використовують чат на основі штучного інтелекту з використанням природної мови [1].

Постачальники інтелектуальних цифрових помічників наразі пропонують підтримку для розробників, щоб вебсайти могли розробляти модулі плагінів, які дозволяли б інтелектуальним цифровим помічникам загального призначення мати більше доступу до більшої кількості аспектів онлайн-сервісів, якими користуються користувачі.

Сьогодні це не дуже розповсюджене явище, але, ймовірно, стане більш поширеним у міру зростання поширення інтелектуальних цифрових помічників, враховуючи також той факт, що багато онлайн-магазинів також пропонують програми для смартфонів.

У рамках даної кваліфікаційної роботи було поставлено завдання розробити застосунок «інтелектуальний помічник» з використанням хмарних сервісів. Інтелектуальний помічник є ботом, побудованим з використанням методів породжувального штучного інтелекту, який призначений для спілкування з відвідувачами сайту в режимі реального часу.

Цей проект має на меті розробити сучасний та ефективний інструмент для реалізації віртуального продавця-асистента книжкової крамниці. З його допомогою буде здійснюватися регістрація клієнтів, підбір книжок та оформлення замовлення на сайті магазину.

У процесі розробки застосунку будуть використані сучасні технології програмування, включаючи методи штучного інтелекту, що дозволить створити зручний та функціональний інструмент для забезпечення функціонування інтернет-магазину.

Основною метою цієї розробки є створення бота зі зручним та інтуїтивно зрозумілим інтерфейсом, який дозволить користувачам ефективно взаємодіяти з інтелектуальним помічником в режимі онлайн.

Розроблений застосунок буде враховувати сучасні вимоги до функціональності та швидкодії, що забезпечить його ефективну та коректну роботу та приємний досвід користувачів. Очікується, що розроблений інтелектуальний помічник покращить доступність та результативність здійсненння покупок на сайті, сприяючи підвищенню рівня продажів книжкового інтернет-магазину «BukvaUA».

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Інтелектуальний віртуальний помічник (Intelligent Virtual Assistant, IVA) [2] — це помічник у чаті з підтримкою штучного інтелекту (ШІ), який генерує персоналізовані відповіді, поєднуючи аналітику та когнітивні обчислення на основі індивідуальної інформації про клієнта, минулих розмов і місця розташування, використовуючи корпоративну базу знань і людське розуміння. Він є більш просунутим, ніж простий чат-бот, який є автоматизованим, але не підтримує ШІ.

Клієнти очікують, що інтернет-магазини спілкуватимуться з ними за обраним каналом у той час, коли вони виберуть, і швидко вирішатимуть їхні проблеми чи відповідатимуть на запитання.

Впровадження інтелектуальних помічників здійснюється переважно задля економії коштів. Економія досягається в першу чергу завдяки перехопленню викликів: боти можуть обробляти від 30 до 70 відсотків усіх взаємодій [2], залежно від галузі, дозволяючи як зменшити кількість операторів-людей, так і можливість масштабувати сервіс клієнтської підтримки без додавання нових людей. Хоча наразі багато ботів є простими автоматизованими транзакціями, більш просунуті боти є інтелектуальними віртуальними помічниками, які з часом стають розумнішими завдяки штучному інтелекту або машинному навчанню. Крім того, IVA зможе підвищити ефективність співробітників, допомагаючи в навчанні та допомагаючи співробітникам у їхніх відповідях на запити клієнтів, що призведе до ефективності як бюджетів навчання, так і покращення операційних показників.

Крім економії коштів, компанії прагнуть отримати вигоду від збільшення продажів завдяки впровадженням штучного інтелекту, оскільки IVA завчасно рекомендують оптимальні продукти та послуги для кожного

клієнта та кваліфікують потенційних клієнтів для подальшого спостереження, збільшуючи продажі з мінімальними людськими зусиллями.

Нарешті, якщо правильно впровадити IVA, вони можуть покращити загальну взаємодію з клієнтами, пропонуючи цілодобову доступність послуг без потреби наймати додаткових співробітників, забезпечуючи узгоджені оптимальні відповіді на всіх каналах і забезпечуючи присутність багатоканальної підтримки від вебсайтів і програм до колцентрів і електронної пошти.

Удосконалення технологій і готовність клієнтів співпрацювати з IVA/ботами прискорюють їх впровадження. Оскільки 31 відсоток усіх роздрібних торговців планує інвестувати в чат-боти в наступному році, очікується, що глобальний ринок чат-ботів досягне 1,23 мільярда доларів США до 2025 року, а сукупний річний темп зростання становитиме 24 відсотки, згідно з дослідженнями Grand View Research [2]. Gartner прогнозує, що до 2020 року клієнти керуватимуть 85 відсотками своїх ділових відносин без взаємодії з людиною [2], а McKinsey прогнозує, що продуктивність робочої сили IVA дорівнюватиме продуктивності до 140 мільйонів працівників до 2025 року [2].

Клієнти як ніколи охоче взаємодіють із ботами, якщо взаємодія закінчується сприятливо. Дослідження від LivePerson показало, що 33 відсотки споживачів позитивно сприймають чат-ботів, а 48 відсотків ставляться до них нейтрально, якщо їхню проблему було вирішено [2].

Питання для сучасних керівників полягає не в тому, чи використовувати IVA/ботів, а в тому, як розробити та розгорнути їх найбільш ефективно та результативно.

Усвідомлення переваг IVA вимагає правильного проектування та підходу до впровадження. Більшість сьогоднішніх ботів, орієнтованих на клієнта, мають високий рівень сценаріїв і обмежені можливості. Переважна більшість розуміє та відповідає лише на прості запити, такі як втрачений пароль або пошук поширених запитань. Більшість людей не знають, як

сформулювати прості запити або розбити запит на кілька простих кроків, які може зрозуміти бот.

Це призводить до розчарування клієнта, коли клієнт йде «не за сценарієм», що зрештою вимагає від клієнтів зателефонувати або поспілкуватися з людиною, щоб вирішити свою проблему. Додаткове розчарування полягає в тому, що більшість ботів не забезпечують безперебійного переходу до служби підтримки клієнтів — надають лише номер для дзвінка або посилання для чату, після чого клієнт повинен починати все спочатку.

Уникнення цих недоліків означає створення справді гібридної робочої сили, у якій інтелектуальний IVA спрощує роботу, а співробітники взаємодіють із клієнтами за допомогою більш ефективних взаємодій.

Для подолання деяких з окреслених проблем можуть бути використані боти на основі моделей породжувального штучного інтелекту. Розроблений застосунок повинен надавати зручний та ефективний функціонал для спілкування з клієнтами книжкового інтернет-магазину. Інтелектуальний помічник, що розробляється у даній кваліфікаційній роботі, спрямований на обслуговування відвідувачів вебсайту, і призначений спеціально для книжкової крамниці «BukvaUA».

1.2 Аналіз аналогів інтелектуальних помічників книжкових інтернет-магазинів

Перед початком проектування застосунку необхідно провести пошук і аналіз існуючих аналогічних програмних продуктів, що належать до категорії інтелектуальних помічників або чат-ботів, які використовуються у книжковій онлайн-торгівлі. Було розглянуто вебсайти найбільш відвідуваних книжкових інтернет-магазинів України [4]:

1. «Yakaboo» [5]

Наразі книжковий інтернет-магазин Якабу є одним із найбільших в Україні, має власне видавництво Yakaboo Publishing, яке видає український нішевий нон-фікшн та перекладає українською мовою світові нон-фікшн бестселери. В інтернет-магазині yakaboo.ua продається понад 300 000 книг різних жанрів на 71 мові у паперовому, електронному або аудіоформатах. Окрім книг, доступні товари для творчості, настільні ігри, книжкові аксесуари та подарунки.

Сайт магазину не має окремого чат-бота. На головній сторінці сайту присутня кнопка «Написати повідомлення», натиснувши на яку, користувач отримає перелік контактів слубжи підтримки магазину, включаючи телефон, месенджери, електронну пошту, та соціальні мережі (рис. 1.1).

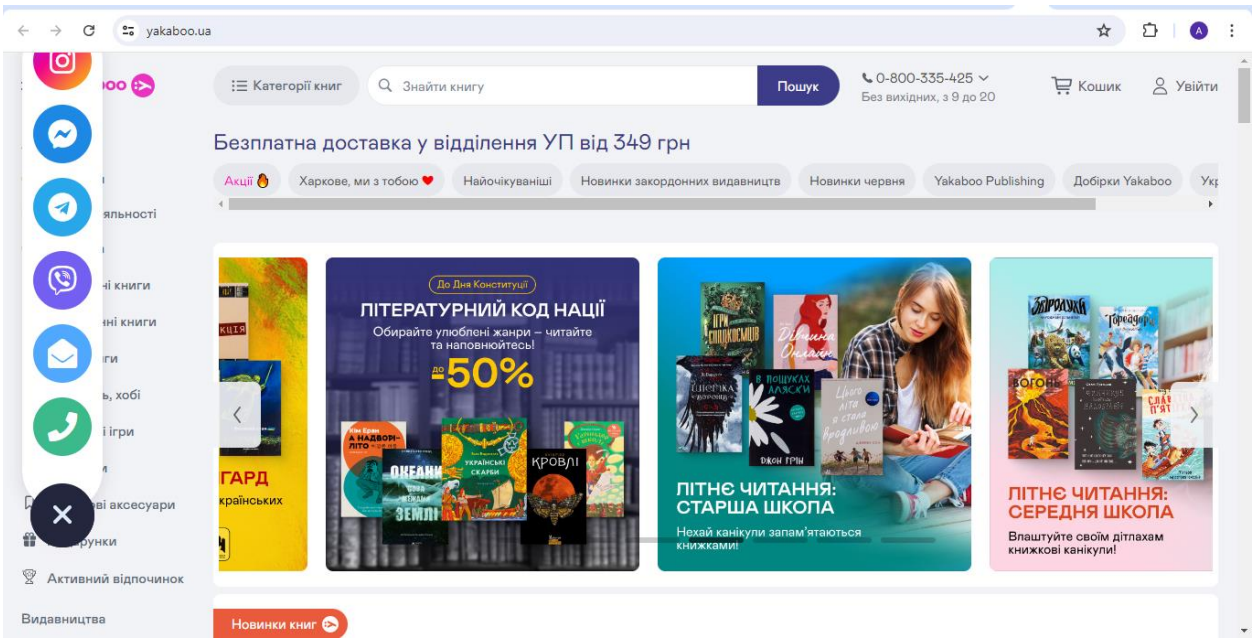

Рис. 1.1. Магазин Yakaboo

При виборі месенджерів Viber або Telegram користувачеві пропонується спілкування з ботом служби підтримки магазину.

2. Книгарня «Є» [6]

В інтернет-магазині book-ye.com.ua можна знайти будь-яку літературу українською мовою для читача від 0 до 99 років, а також книги ексклюзивних та рідкісних видань.

На сторінках вебсайту немає чату для спілкування з ботом або оператором контакт-центру, наявні лише телефони у шапці сайту, та кнопка для дозволу на отримання сповіщень (рис. 1.2).

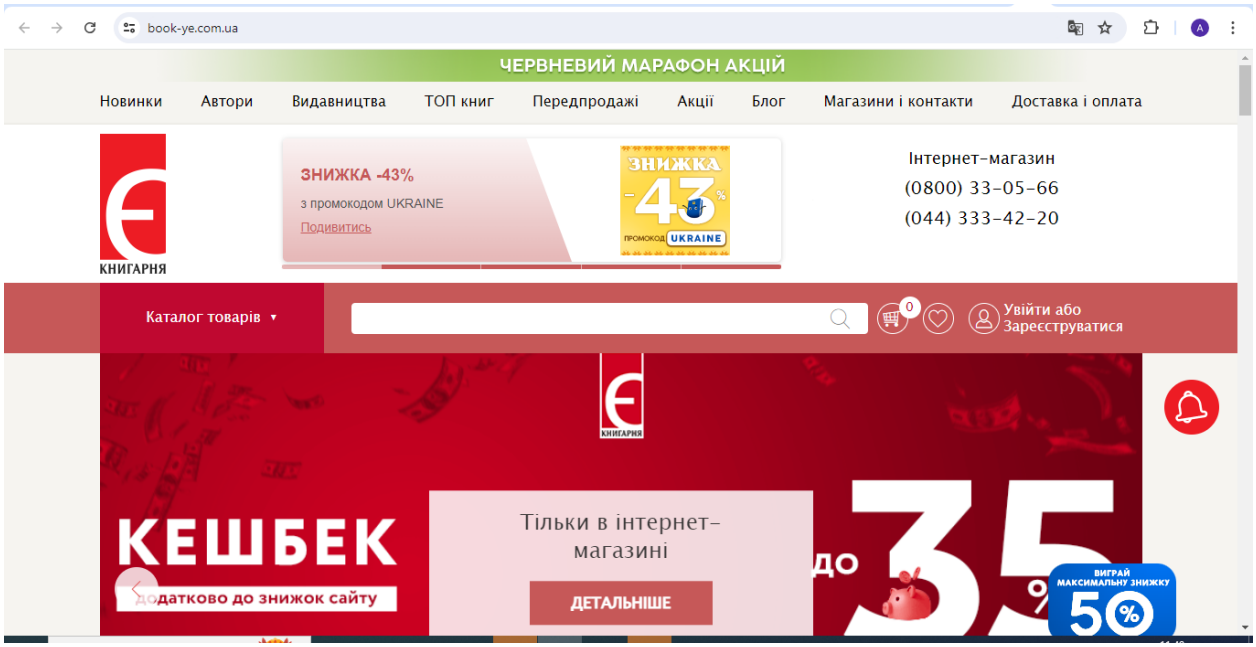

Рис. 1.2. Вебсайт книгарні «Є»

3. «Клуб Сімейного Дозвілля» [7]

Книжковий Клуб «Клуб Сімейного Дозвілля» є одним із найбільших книжкових дистриб'юторів в Україні, має 38 філій по всій країні. Щороку до війни видавництво Клубу видавало близько 600 найменувань художньої, дитячої та прикладної літератури. Публікує як українських, світових зіркових авторів, так і твори молодих українських письменників.

На сайті даного інтернет-магазину також відсутні інструменти для онлайн-спілкування з менеджером, або чат-бот (рис. 1.3).

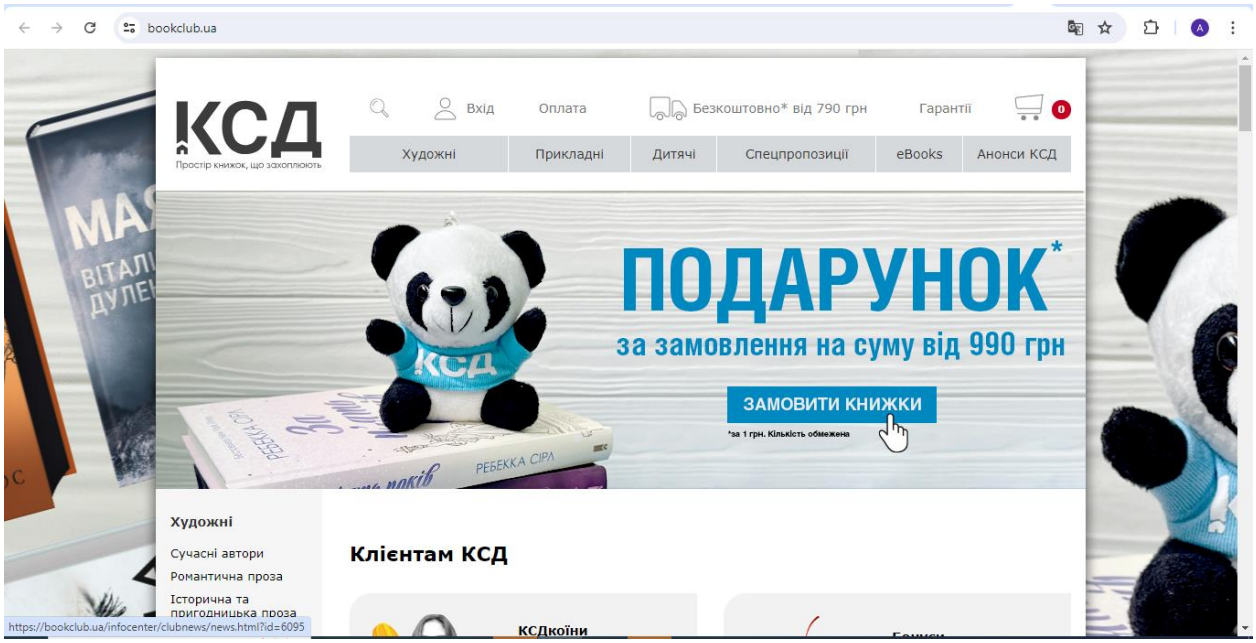

Рис. 1.3. «Клуб Сімейного Дозвілля»

4. «Bookovka» [8]

Сучасний інтернет-магазин bookovka.ua пропонує понад 1 млн книг та інших товарів: комікси, електронні видання, товари для творчості й активного відпочинку, настільні ігри, канцтовари, дитячі товари.

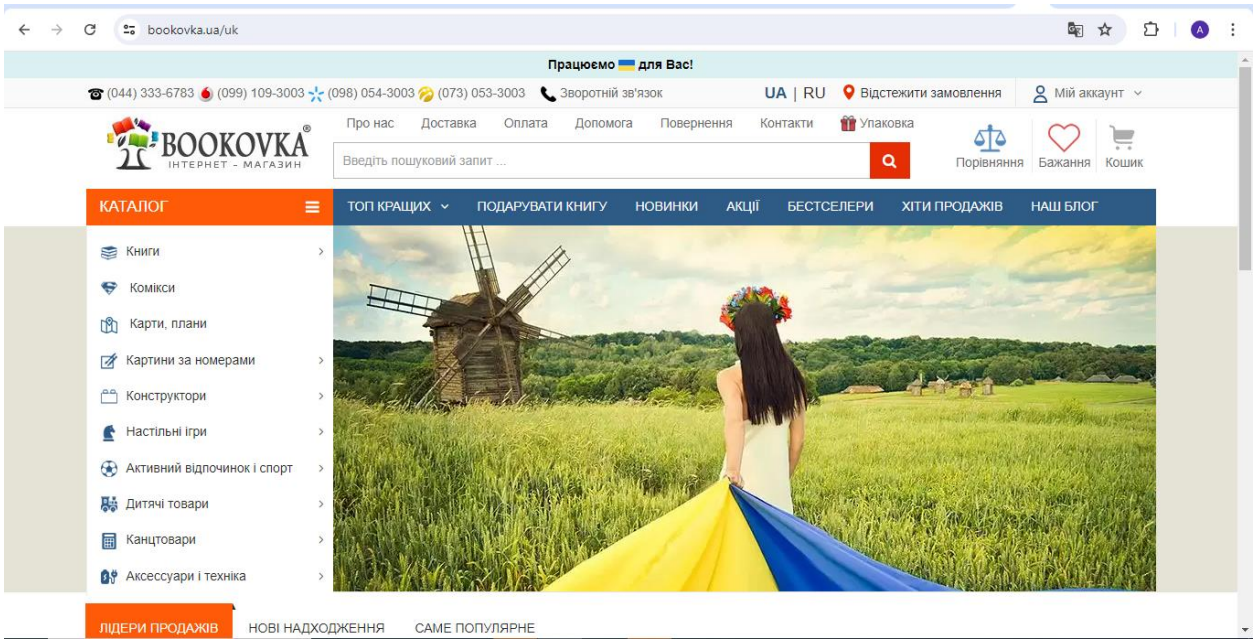

Рис. 1.4. Магазин «Bookovka»

На цьому сайті, як і попередніх, інструменти чату відсутні. У шапці сайту наведені номери телефонів кол-центру, та кнопка замовлення дзвінка (рис. 1.4).

5. «Book24» [9]

Це молодий інтернет-маркет книжкової продукції в Україні, який стрімко розвивається. Пропонує до продажу понад 10 тисяч найменувань книг, а також ігри та електронні формати книг.

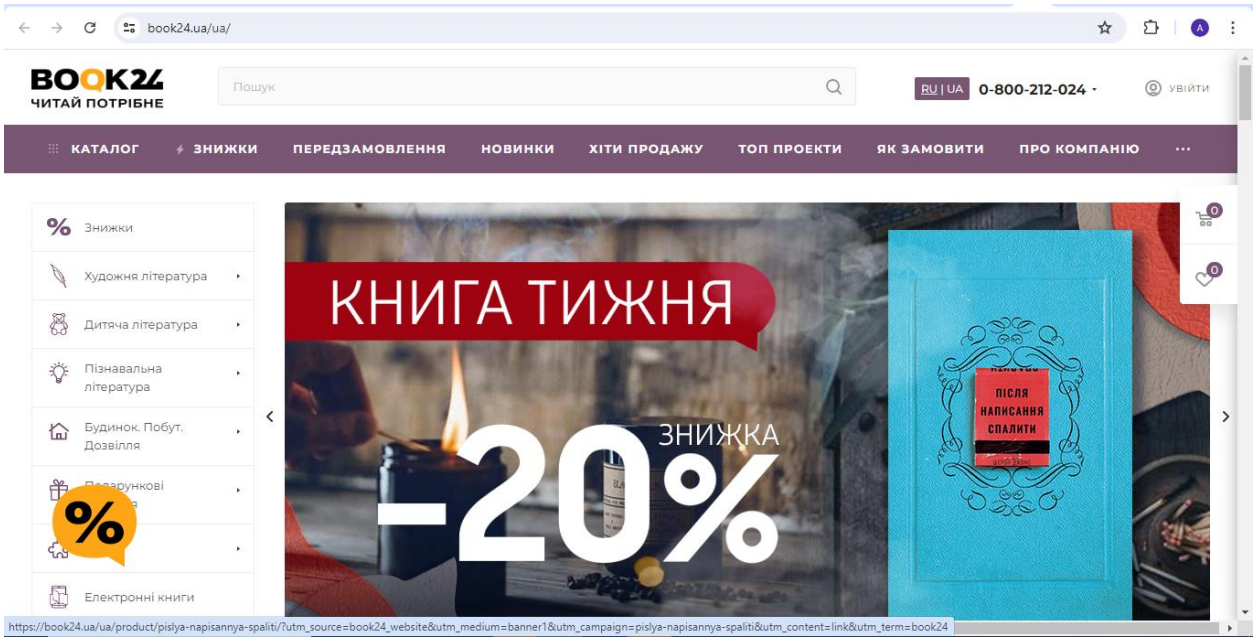

Рис. 1.5. Магазин «Book24»

На сайті даного магазину наявний лише номер телефону, та кнопка для оформлення підписки на знижки.

6. «Balka-book» [10]

Харківський книжковий інтернет-магазин balka-book.com заснований у 2004 році та спеціалізується на продажі комп'ютерної, технічної, бізнес та медичної літератури. Це робить інтернет-магазин унікальним на ринку книжкової індустрії. В асортименті можна знайти також книги художньої, навчальної, юридичної, економічної та дитячої літератури, вінілові пластинки, касети, CD та DVD диски.

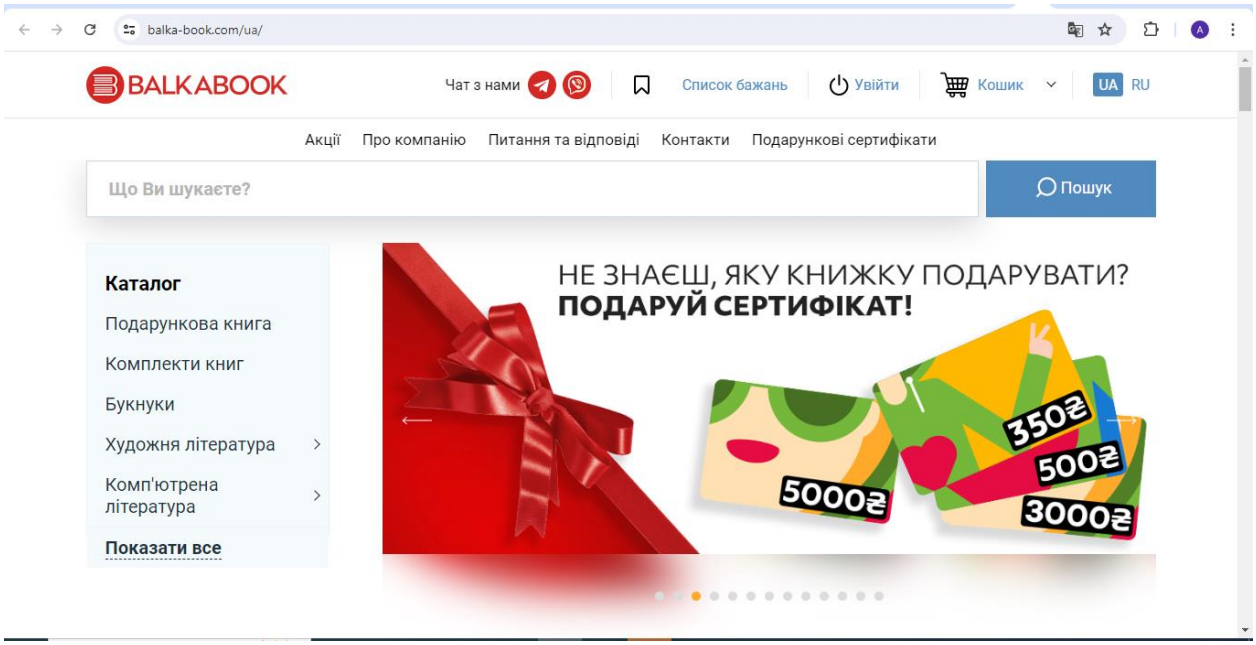

Рис. 1.6. Магазин «Balka-book»

На рис. 1.6 можна побачити, що на сайті магазину є кнопки для початку спілкування з чат-ботами у Viber та Telegram.

7. «Лавка Бабуїн» [11]

Книжковий інтернет-магазин України, де можна замовити книги у всіх жанрах, сувеніри та подарунки, зокрема: дитячу, художню, науково-популярну, бізнес-літературу, фотоальбоми, артбуки, іграшки, товари для творчості, канцелярію та іншу продукцію.

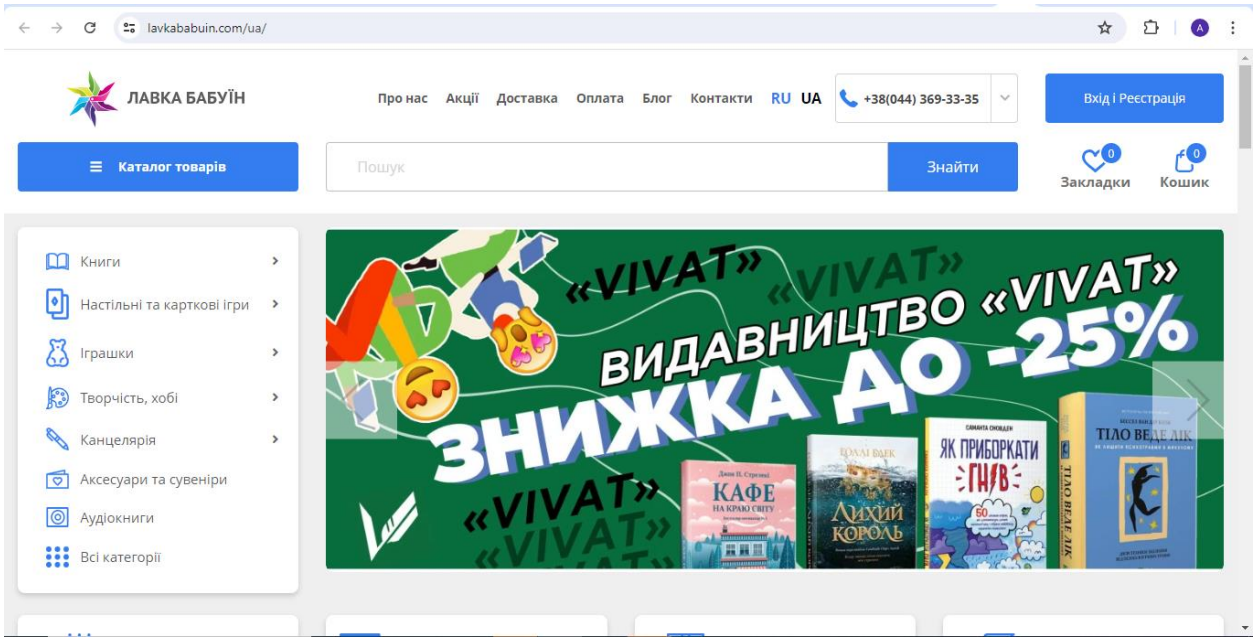

Рис. 1.7. Магазин «Лавка Бабуїн»

На сайті цього інтернет-магазину також відсутні чати або інтелектуальні помічники.

8. «Booklya» [12]

«Букля» — книжковий інтернет-магазин для всієї родини, котрий активно розвивається. Каталог налічує 210 тис. найменувань книг різних жанрів. Окрім книг, в магазині продаються дитячі товари, настільні ігри, конструктори LEGO, товари для хобі та творчості, оригінальні блокноти й планери.

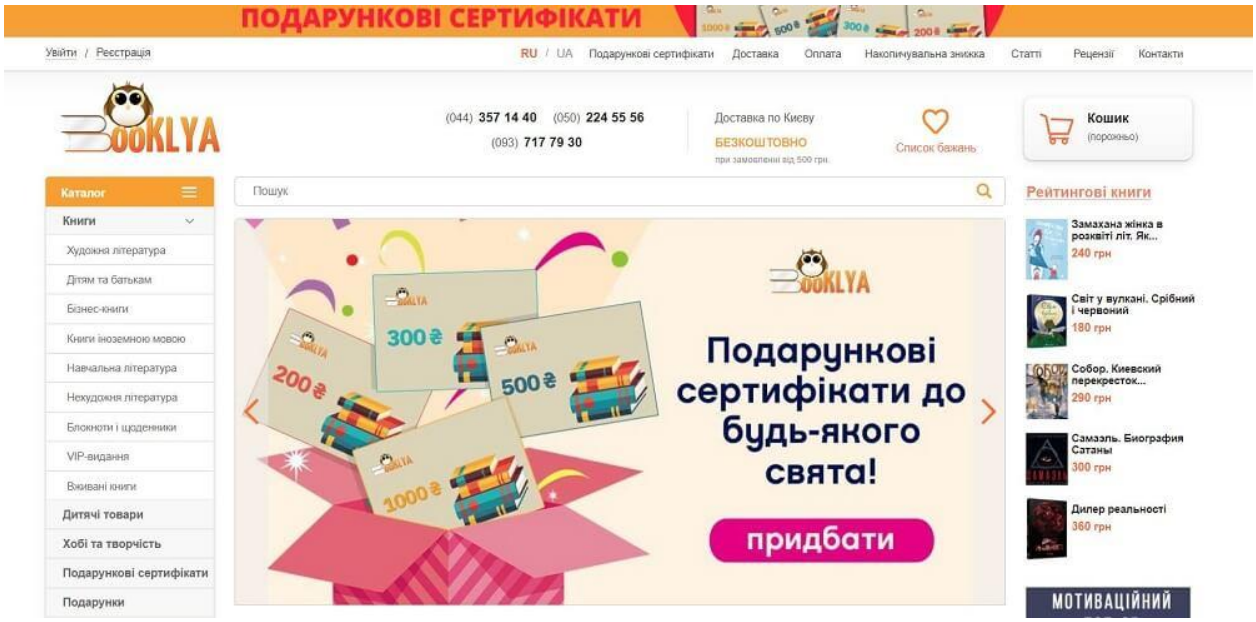

Рис. 1.8. Сайт магазину «Booklya»

На сторінках даного магазину є лише контактні телефони, вони розміщені у шпапці сайту.

Проведений огляду показав, що лише два із восьми найбільших книжкових інтернет-магазинів України мають на своїх сайтах інструменти для спілкування з ботами, до того ж вони реалізовані через стронні месенджери, а не інтегровані у вебзастосуток сайту. Тому актуальною задачею є розробка інтелектуального помічника для сайту книжкового магазину «BukvaUA», який би реалізовував функції віртуального асистента для консультування

відвідувачів вебсайту, та супроводу процесів реєстрації та оформлення замовлення.

1.3 Мета та задачі застосунку інтелектуального помічника

Мета даної кваліфікаційної роботи полягає у розробці програмного забезпечення інтелектуального помічника з використанням хмарних сервісів. Цей застосунок реалізує зручний та функціональний інструмент для забезпечення функціонування чат-бота на сайті книжкового інтернет-магазину. Він покликаний сприяти спрощенню та ефективності обслуговування відвідувачів сайту, та підвищенню рівня продажів інтернет-магазину.

Також, враховуючи основну мету цієї програми, вебзастосунок «інтелектуальний помічник» буде використовуватись у якості віртуального асистента, який консультує відвідувачів вебсайту, та супроводжує процеси реєстрації та оформлення замовлення.

Розроблений застосунок має забезпечувати виконання наступних функцій:

- легка навігація, простий дизайн;
- адаптивний дизайн для різних пристроїв;
- перегляд ассортименту книг;
- можливість замовлення книги;
- перегляд вмісту кошика з поточним замовленням;
- можливість зміни вмісту кошика з поточним замовлення;
- отримання інтелектуальної поради щодо добору книжок;
- отримання книги щодо фабули книги;
- отримання відповіді щодо сюжету книги у розгорнутому вигляді;
- можливість включення посилань на книжки у контекст відповіді онлайн помічника.

Застосунок інтелектуального помічника має бути розроблений відповідно до сучасних вимог до функціональності та швидкодії. Програмне забезпечення повинно підтримувати виконання таких дій:

- обмін HTTP-запитами з вебсервером інтернет-магазину;
- обмін запитами із базою даних сайту;
- взаємодія із хмарним сервером Google за допомогою API обраної моделі ШІ;
- генерація діалогів;
- підтримка багатьох мов.

Для досягнення поставленої мети і відповідно до вимог, необхідно виконати наступні завдання:

– здійснити експертний аналіз предметної області;

– вибрати необхідні програмні інструменти для створення застосунку;

– розробити артітектуру програмного продукту;

– розробити відповідний функціонал застосунку;

– провести тестування застосунку.

Щоб забезпечити ефективну реалізацію проекту та виконання всіх вимог, необхідно мати детальне технічне завдання. Це є необхідною умовою для успішної реалізації застосунку та досягнення поставлених цілей.

1.4 Висновки до розділу 1

У першому розділі кваліфікаційної роботи бакалавра було розглянуто структуру та особливості функціонування інтелектуальних віртуальних помічників, а також виконано аналіз популярних книжкових інтернет-магазинів на предмет провадження віртуальних асистентів.

В результаті огляду була виявлена обмежена кількість випадків впровадження інтелектуальних помічників у вебсайти книжкових крамниць, здебільшого у якості чат-ботів сторонніх месенджерів.

Вищезазначена сутиація обумовлює актуальність розробки інтелектуального помічника з використанням хмарних сервісів, який би покращував загальну взаємодію з відвідувачами та забезпечував цілодобову доступність послуг клієнтської підтримки книжкового інтернет-магазину.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ЗАСТОСУНКУ ІНТЕЛЕКТУАЛЬНОГО ПОМІЧНИКА

2.1 Варіанти використання застосунку

Ця діаграма використання ілюструє, як користувач взаємодіє з системою книжкового магазину. Всі випадки використання пов'язані з основними функціями системи, що забезпечують зручний процес вибору та замовлення книг для користувача. Штучний асистент допомагає користувачу з рекомендаціями, що підвищує задоволеність клієнтів та покращує їхній досвід взаємодії із системою.

Рис. 2.1. Діграма варіантів використання застосунку

Актори:

- Користувач: Це кінцевий користувач, який взаємодіє із системою для перегляду, вибору та замовлення книг.

Варіанти використання (Use Cases):

1. Перегляд сторінки книги:

- Опис: Користувач може переглядати деталі книги, такі як назва, автор, опис, ціна на її сторінці.

- Ціль: Дати користувачу можливість отримати повну інформацію про книгу перед її покупкою.

2. Додати до кошика книгу:

- Опис: Користувач може додати обрану книгу до свого кошика для подальшого замовлення.

- Ціль: Дозволити користувачу зібрати всі бажані книги перед оформленням замовлення.

3. Видалити з кошика:

- Опис: Користувач може видалити книгу з кошика, якщо передумав її купувати.

- Ціль: Дати користувачу можливість керувати своїм кошиком, видаляючи непотрібні книги.

4. Отримання поради щодо добору книги:

- Опис: Користувач може отримати рекомендації від штучного асистента щодо вибору книги на основі його уподобань.

- Ціль: Надати користувачу рекомендації для полегшення вибору книги.

5. Замовити книги:

- Опис: Користувач може оформити замовлення на книги, які додані до кошика.

- Ціль: Завершити процес покупки, забезпечуючи користувачу можливість отримати обрані книги.

Зв'язки між акторами та випадками використання:

Користувач взаємодіє з усіма випадками використання:

- переглядає сторінку книги;
- додає книгу до кошика;
- видаляє книгу з кошика;
- отримує поради щодо вибору книги;
- замовляє книги.

2.2 Вибір технологій та мов програмування для створення програмного продукту

Для створення застосунку були обрані фреймворк Flask, мова програмування Python, СУБД MySQL та модель штучного інтелекту Gemini AI.

Flask [13] — мікрофреймворк для вебдодатків, створений з використанням Python. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2. Поширюється відповідно до умов ліцензії BSD. Flask використовується для розробки таких проєктів як Pinterest, LinkedIn, а також сторінка спільноти Flask.

Flask називається мікрофреймворком, оскільки він не вимагає спеціальних засобів чи бібліотек. У ньому відсутній рівень абстракції для роботи з базою даних, перевірки форм або інші компоненти, які надають широковживані функції за допомогою сторонніх бібліотек. Однак, Flask має підтримку розширень, які надають додаткові властивості таким чином, наче вони були доступні у Flask із самого початку. Існують розширення для встановлення об'єктно-реляційних зв'язків, перевірки форм, контролю процесу завантаження, підтримки різноманітних відкритих технологій аутентифікації та декількох поширених засобів для фреймворку. Розширення оновлюються частіше, аніж базовий код.

Основні властивості Flask:

- містить сервер для розробки та відлагоджувач;
- вбудована підтримка юніт-тестів;
- управління запитами RESTful;
- використовує шаблони Jinja2;
- має підтримку безпечних куків (сесії на стороні клієнта);
- 100% відповідність WSGI 1.0;
- підтримка Unicode;
- докладна документація;

- сумісність з Google App Engine;
- наявність розширень для забезпечення бажаної поведінки.

API інтелектуального чат-бота [14], який також називають автоматичними ботами для обміну повідомленнями, — це інтерфейс програмування, який дозволяє розробникам включати функції чат-бота у свої програми, веб-сайти або служби. Ці системи, також відомі як Conversational AI API, дозволяють розробникам отримувати доступ до можливостей обробки природної мови (NLP) попередньо навчених мовних моделей або сервісів чат-ботів для виконання NLP-дій і створення відповідей на запити користувачів.

Вони можуть інтерпретувати введені користувачем дані, виявляти наміри та генерувати осмислені відповіді. Розробники можуть використовувати програмні інтерфейси прикладних програм (API) AI Chatbot для створення програм, які можуть аналізувати та відповідати на запитання користувачів за допомогою природної мови. Ці API часто використовують масивні мовні моделі для створення людських відповідей.

Розробники можуть інтегрувати генеративні чат-боти зі штучним інтелектом в існуючі платформи або додатки, використовуючи API, надані постачальниками ШІ, такими як NLP Cloud, які забезпечують безперебійний зв'язок між чат-ботом і бекендом платформи. Це передбачає надсилання користувацьких даних штучному інтелекту через API, отримання згенерованої штучним інтелектом відповіді та її представлення через користувацький інтерфейс програми.

Чат-боти на основі генеративного ШІ можна широко налаштовувати відповідно до конкретних потреб бізнесу, включаючи адаптацію їхніх відповідей, тональності і навіть бази знань, з якої вони черпають інформацію, що робить їх дуже універсальними для різних галузей і застосувань.

З метою вибору оптимальної моделі ШІ для інтелектуального помічника був проведений порівняльний аналіз чотирьох сучасних моделей ШІ з API інтелектуального чат-бота. Результати аналізу представлені у таблиці 2.1.

Таблиця 2.1 – Порівняння платформ ШІ

Критерій	Gemini AI [15]	OpenAI (ChatGPT) [16]	Dialog Flow [17]	NLP (наприклад, spaCy [18])
Функціональ-ність	Висока, широкий спектр можливостей аналізу тексту та обробки запитів	Висока, потужний генеративний ШІ для діалогів	Висока, підтримка складних діалогів та дій	Висока, потужний інструмент для аналізу тексту
Простота інтеграції	Середня, потребує налаштування та API	Середня, потребує роботи з API	Висока, готові інтеграції з різними платформами	Середня, потребує налаштування та інтеграції через код
Підтримка мов	Багатомовна підтримка	Багатомовна підтримка	Багатомовна підтримка	Обмежена, залежить від використаної бібліотеки
Якість обробки мови	Висока	Дуже висока, природні відповіді	Висока	Висока, залежить від моделі
Можливості удосконалення	Високі, гнучке налаштування	Високі, можна навчати на специфічних даних	Високі, налаштування діалогових моделей	Високі, повний контроль над моделлю
Вартість	Є безкоштовна моделі	Немає безкоштовної моделі. Вартість висока.	Середня, різні тарифні плани	Здебільшого безкоштовна.
Спільнота та документація	Середня, обмежена спільнота	Висока, велика спільнота та підтримка	Висока, активна спільнота та підтримка	Висока, велика спільнота розробників

Виходячи з результаті аналізу, вибір був зроблений на користь Gemini AI. Вибір платформи Gemini API AI має кілька важливих переваг, які роблять її привабливим вибором для інтеграції та використання:

1. Функціональність:

Висока функціональність: Gemini API AI пропонує широкий спектр можливостей аналізу тексту та обробки запитів. Це дозволяє обробляти різні види текстових даних ефективно та точно.

2. Простота інтеграції:

Середня складність: хоча інтеграція вимагає налаштування та роботи з API, це забезпечує гнучкість у налаштуванні системи під специфічні потреби.

3. Підтримка мов:

Багатомовна підтримка: це дозволяє використовувати платформу для обробки текстів різними мовами, що є великою перевагою для глобальних застосувань.

4. Якість обробки мови:

Висока якість: Gemini API AI забезпечує високу якість обробки мови, що важливо для точності та природності відповідей.

5. Можливості кастомізації:

Високі можливості кастомізації: можна налаштовувати модель під специфічні потреби та дані, що дозволяє створювати рішення, оптимізовані під конкретні задачі.

6. Вартість:

Наявність безкоштовної моделі: це дозволяє розпочати роботу з платформою без значних фінансових витрат, що є перевагою для невеликих проектів або стартапів.

7. Спільнота та документація:

Середня спільнота та документація: хоча спільнота може бути обмежена порівняно з великими платформами, доступ до документації та підтримки дозволяє розв'язувати більшість питань, які можуть виникнути під час роботи з платформою.

Отже, Gemini API AI поєднує в собі високу функціональність, гнучкість у налаштуванні, багато мовну підтримку та високу якість обробки тексту, що робить її привабливим вибором для широкого спектру застосувань.

В якості системи управління базою даних вибір був зроблений на користь на MySQL [19], оскільки вона має такі переваги:

1. Надійність та продуктивність: MySQL відомий своєю високою продуктивністю, стабільністю та надійністю. Він здатний обробляти великі обсяги даних і високі навантаження.

2. Підтримка та документація: MySQL має велику базу користувачів і чудову документацію. Це означає, що для більшості проблем можна знайти рішення в інтернеті.

3. Сумісність з іншими інструментами: MySQL добре інтегрується з різними мовами програмування та інструментами, що полегшує розробку і підтримку додатків.

2.3 Архітектура інтелектуального помічника

На високому рівні абстракції чат-бот на основі Gemini AI має архітектуру [20], яка складається з трьох основних компонентів (рис. 2.2): компонент NLU, менеджер діалогу, та генератор повідомлень.

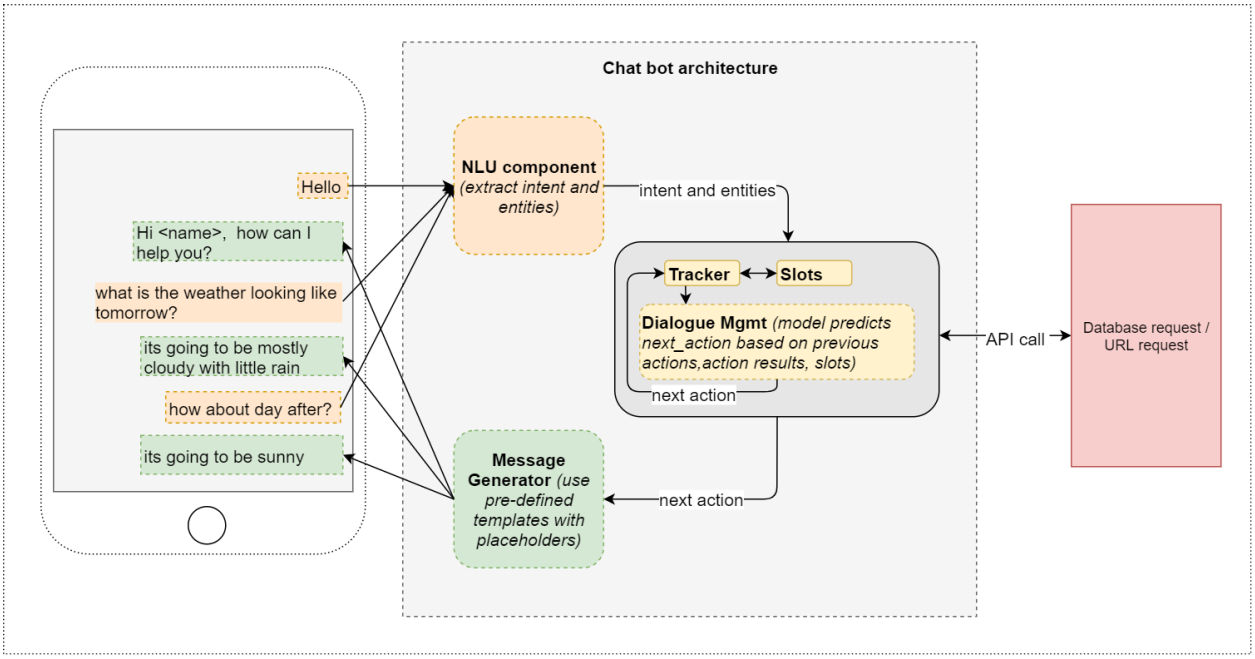

Рис. 2.2. Архітектура ботаGemini AI [20]

Компонент NLU складається з:

- Контрольована модель класифікації намірів, яка тренується на різноманітних реченнях як вхідних даних і намірах як цілях. Як правило, лінійної SVM буде достатньо як модель класифікації намірів.
- Модель вилучення сутності — це може бути попередньо навчена модель, як-от бібліотека Spacy або StanfordNLP, її можна навчити за допомогою деяких імовірнісних моделей, як-от CRF (умовні випадкові поля).

Оскільки інтелектуальний помічник є розмовним ботом, потрібно відстежувати розмови, які відбулися до поточного часу, щоб передбачити відповідну відповідь. Для цього потрібен об’єкт словника, який можна зберігати з інформацією про поточний намір, поточні сутності, постійну інформацію, яку користувач надав би попереднім запитанням бота, попередню дію бота, результати виклику API. Ця інформація становитиме вхідний X, вектор ознак. Ціль, на якій навчатиметься модель діалогу, буде «next_action», «наступною дією» («наступна дія» може бути просто закодованим вектором, що відповідає кожній дії, яку ми визначаємо в наших навчальних даних).

Отримання навчальних значень для вектора ознак X, тобто інформації про наміри та сутності, здійснюється з компонента NLU.

Для того, щоб отримати решту значень (інформацію, яку користувач надав би попереднім запитанням бота, попередню дію бота, результати виклику API тощо) використовується компонент менеджера діалогу. Ці значення функцій беруться з навчальних даних, які користувач визначить у формі зразків розмов між користувачем і ботом. Ці зразки розмов повинні бути підготовлені таким чином, щоб вони охоплювали більшість можливих потоків розмов, прикидаючись одночасно користувачем і ботом.

Після отримання всіх необхідних значень функцій із зразків бесід у потрібному форматі, виконується навчання моделі штучного інтелекту, щоб передбачити next_action. Як видно на рисунку 2.2, це те, що робить компонент «керування діалогом».

Прогнозоване значення next_action може бути приблизно таким:

- відповісти користувачеві відповідним повідомленням;
- отримати дані з бази даних. Після цього робиться виклик API та отримуються результати, які відповідають наміру.

Якщо це буде виклик API / отримання даних, тоді дескриптор потоку керування залишиться в межах компонента «керування діалогом», який надалі використовуватиме/збереже цю інформацію для прогнозування next_action ще раз. Менеджер діалогу оновить свій поточний стан на основі цієї дії та отриманих результатів, щоб зробити наступний прогноз. Коли next_action становить «відповідь користувачеві», підключається компонент «генератор повідомлень».

Компонент генератора повідомлень складається з кількох визначених шаблонів, які відображаються у іменах дій. Таким чином, залежно від дії, передбаченої диспетчером діалогу, викликається відповідне шаблонне повідомлення. Якщо шаблон вимагає заповнення деяких значень-заповнювачів, ці значення також передаються менеджером діалогу генератору. Потім користувачеві відображається відповідне повідомлення, і бот переходить у режим очікування, очікуючи введення користувача.

Взаємодія чат-бота з вебзастосунком сайта відбувається шляхом виклику функцій. Виклик функцій дає змогу розробникам описувати функції (відомі також як інструменти), і змушувати модель інтелектуально вибирати об’єкт JSON, що містить аргументи для виклику цих функцій. Це дозволяє:

- Автономне прийняття рішень: моделі можуть розумно вибирати інструменти, щоб відповідати на запитання.
- Надійний синтаксичний аналіз: відповіді надаються у форматі JSON замість більш типової відповіді, схожої на діалог. Саме це дозволяє LLM підключатися до зовнішніх систем, скажімо, через API зі структурованими вхідними даними.

Автономні помічники зі штучним інтелектом можуть взаємодіяти з внутрішніми системами для таких завдань, як замовлення клієнтів і реєстрація, окрім надання відповідей на запити.

Згідно з документацією Gemini AI [21], виклик функцій має наведену на рисунку 2.3 структуру.

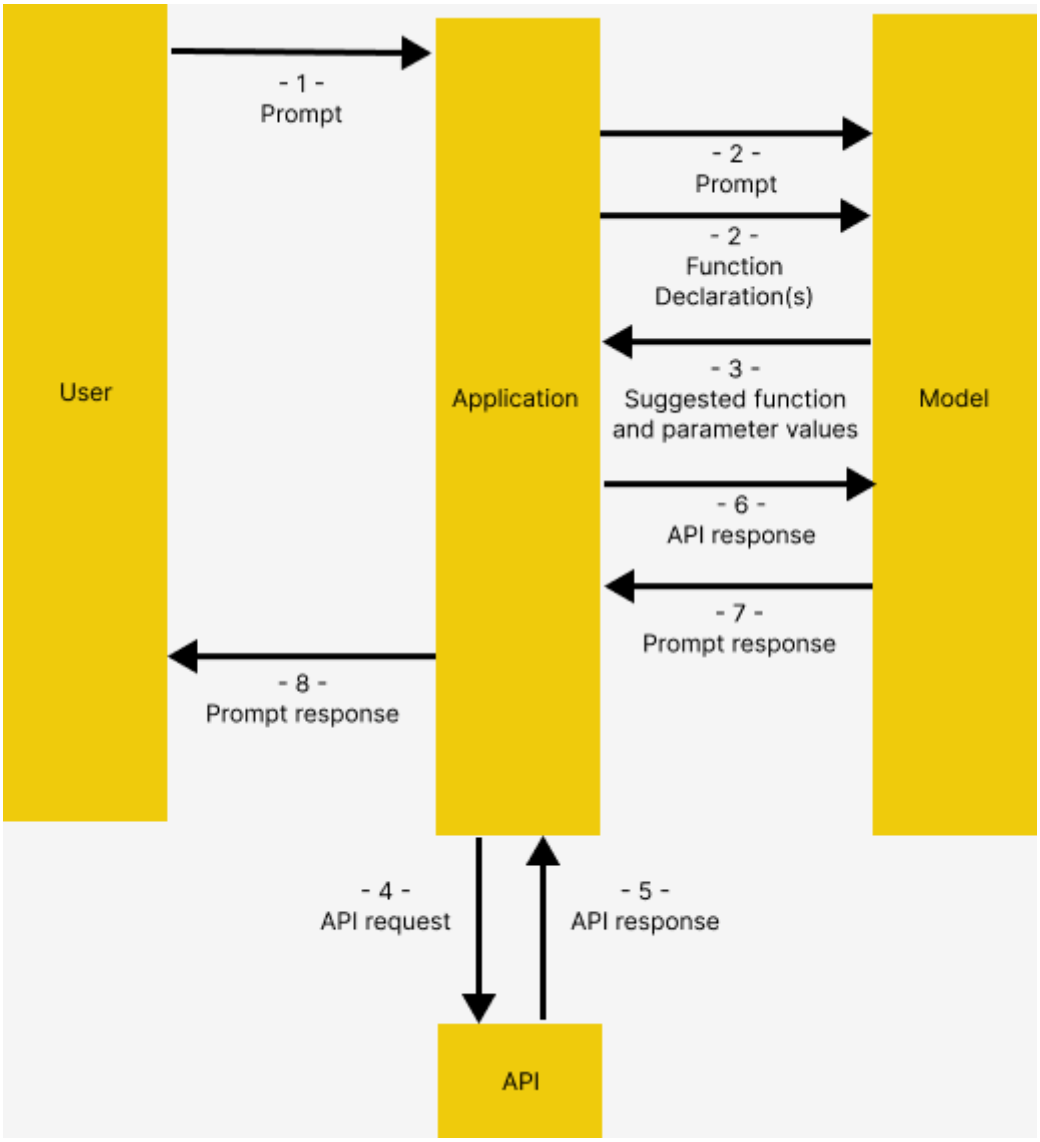

Рис. 2.3. Процес виклику Gemini API AI як функцій

Процес виклику Gemini API AI як функцій включає наступні кроки:

1. Підказка про проблеми користувача в програмі.

2. Програма передає надану користувачем підказку та декларацію(и) функції, яка є описом інструменту(ів), який може використовувати модель.

3. Базуючись на декларації функції, модель пропонує інструмент для використання та відповідні параметри запиту. Модель виводить лише запропонований інструмент і параметри, без виклику функцій.

4-5. На основі відповіді програма викликає відповідний API

6-7. Відповідь від API знову подається в модель для виведення зрозумілої людині відповіді.

8. Програма повертає остаточну відповідь користувачеві, потім повторює з кроку 1.

Можна використовувати виклик функцій, щоб визначити спеціальні функції та надати їх генеративній моделі ШІ.

Під час обробки підказки модель може делегувати певні завдання обробки даних функціям, які визначає розробник. Модель не викликає функції безпосередньо. Натомість модель забезпечує структурований вихід даних, який включає функцію для виклику та значення параметрів для використання.

Можна використовувати структурований вихід із моделі для виклику зовнішніх API. Потім вихідні дані API надаються моделі назад, дозволяючи їй завершити відповідь на підказку.

Згідно результатів аналізу предметної області була спроектована база даних фізична схема якої наведена на рисунку 2.4.

Схема бази даних складається з чотирьох таблиць: customer, genre, book, та books_order. Кожна з цих таблиць має свої стовпці і взаємозв'язки. Нижче наведено детальний опис кожної таблиці та їх атрибутів.

Таблиця customer (клієнт)

- id: Ціле число (INT), автоматично збільшується (AUTO_INCREMENT), первинний ключ (PRIMARY KEY).
- name: Строка (VARCHAR) довжиною до 255 символів, ім'я клієнта.
- phone: Велике ціле число (BIGINT), номер телефону клієнта.
- login: Строка (VARCHAR) довжиною до 255 символів, логін клієнта.

- password: Строка (VARCHAR) довжиною до 255 символів, хешований пароль клієнта.

Таблиця genre (жанр)

- id: Ціле число (INT), автоматично збільшується (AUTO_INCREMENT), первинний ключ (PRIMARY KEY).
- genre_name: Строка (VARCHAR) довжиною до 255 символів, назва жанру.

Таблиця book (книга)

- id: Ціле число (INT), автоматично збільшується (AUTO_INCREMENT), первинний ключ (PRIMARY KEY).
- book_name: Строка (VARCHAR) довжиною до 255 символів, назва книги.
- book_author: Строка (VARCHAR) довжиною до 255 символів, автор книги.
- price: Десяткове число (DECIMAL) з точністю до 10 знаків, з яких 2 після коми, ціна книги.
- genre_book: Ціле число (INT), не може бути NULL, зовнішній ключ (FOREIGN KEY) на таблицю genre.
- image: Строка (VARCHAR) довжиною до 255 символів, URL зображення книги.

Таблиця books_order (замовлення книг)

- id: Ціле число (INT), автоматично збільшується (AUTO_INCREMENT), первинний ключ (PRIMARY KEY).
- customer_id: Ціле число (INT), не може бути NULL, зовнішній ключ (FOREIGN KEY) на таблицю customer.
- book_id: Ціле число (INT), не може бути NULL, зовнішній ключ (FOREIGN KEY) на таблицю book.

Взаємозв'язки між таблицями:

1. Таблиця book має зовнішній ключ genre_book, який посилається на стовпець id в таблиці genre. Це означає, що кожна книга належить до певного жанру.

2. Таблиця books_order має два зовнішні ключі: customer_id та book_id, які посилаються відповідно на стовпці id в таблицях customer і book. Це означає, що кожне замовлення пов'язане з конкретним клієнтом і конкретною книгою.

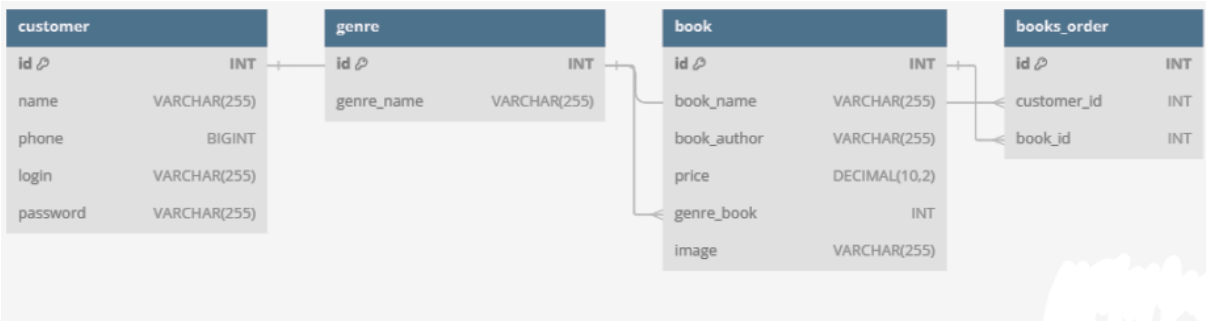

Рис. 2.4. Фізична модель бази даних

Ця структура бази даних дозволяє ефективно зберігати і обробляти інформацію про клієнтів, жанри, книги і замовлення, а також забезпечує цілісність даних через використання зовнішніх ключів.

Детальна документація бази даних представлена у таблицях 2.2-2.5.

Таблиця 2.2 – Поля таблиці customer

Назва стовпчика	Тип даних	Призначення
id	INT	Первинний ключ клієнта
name	VARCHAR (255)	Ім’я клієнта
phone	BIGINT	Телефонний номер клієнта
login	VARCHAR (255)	Логін клієнта
password	VARCHAR (255)	Пароль клієнта

Таблиця 2.3 – поля таблиці genre

Назва стовпчика	Тип даних	Призначення
id	INT	Первинний ключ жанру
genre_name	VARCHAR (255)	Назва жанру

Таблиця 2.4 – поля таблиці book

Назва стовпчика	Тип даних	Призначення
id	INT	Первинний ключ книги
book_name	VARCHAR (255)	Назва книги
book_author	VARCHAR (255)	Автор книги
price	DECIMAL (10, 2)	Ціна книги
genre_book	INT	Жанр книги (зовнішній ключ на genre.id)
image	VARCHAR(255)	Назва зображення книги

Таблиця 2.5 – поля таблиці books_order

Назва стовпчика	Тип даних	Призначення
id	INT	Первинний ключ замовлення
customer_id	INT	ID клієнта (зовнішній ключ на customer.id)
book_id	INT	ID книги (зовнішній ключ на book.id)

2.4 Висновки до розділу 2

У другому розділі кваліфікаційної роботи було проведено проєктування інтелектуального помічника. Спочатку були розглянуті варіанти використання застосунку.

Для створення програмного коду застосунку було обрано фреймворк Flask та мову програмування Python. У якості моделі ШІ була обрана Gemini API AI, перевагами якої є висока функціональність, помірна складність, багатомовна підтримка, та високі можливості кастомізації. У якості СУБД була використана MySQL.

Було наведено опис архітектури чат-бота, та пояснено процес виклику API моделі Gemini. Також була наведена докладна документація бази даних застосунка.

РОЗДІЛ 3. ОПИС ПРОГРАМНОГО КОМПОНЕНТУ ІНТЕЛЕКТУАЛЬНОГО ПОМІЧНИКА

3.1 Архітектура вебзастосунку

Спираючись на матеріали попередніх розділів було розроблено веб застосунок що реалізує адміністративні функції книжкового магазину з вбудованим помічником. Зображена на рисунку 3.1 схема демонструє запити в типовому веб-застосунку, що складається з користувача, веб-сервера, Flask-застосунку, бази даних та зовнішнього сервісу (Google сервер).

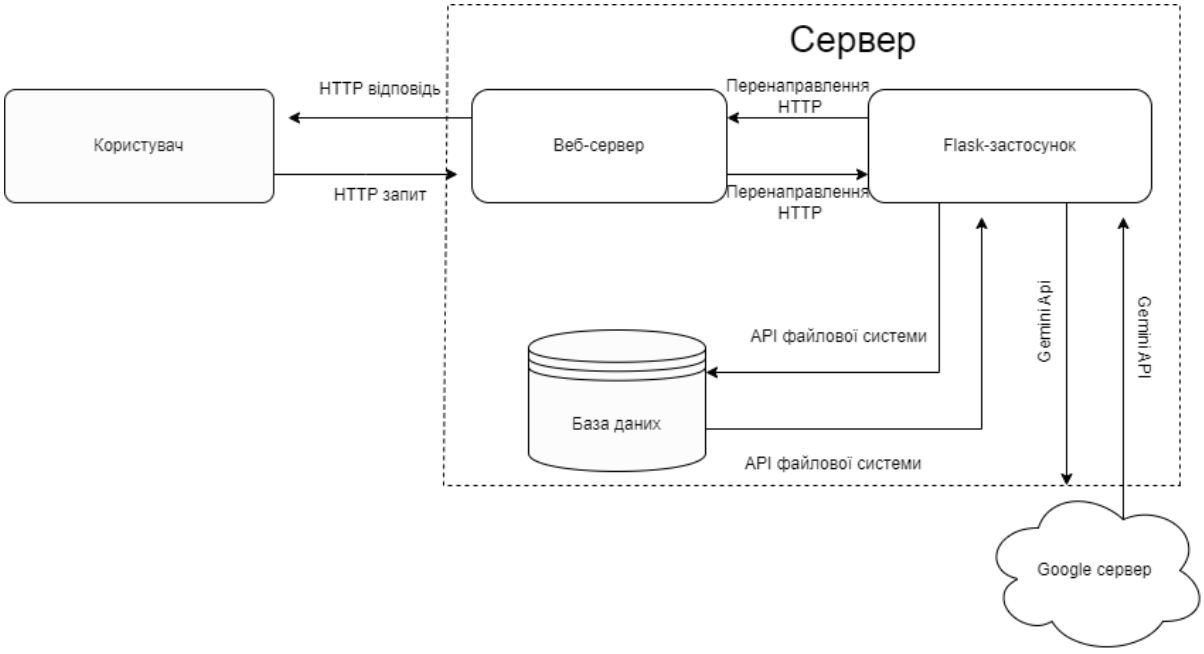

Рис. 3.1. Архітектура вебзастосунку

Ось покроковий опис маршрутизації запитів:

1. Користувач (Користувач -> Веб-сервер):

- Користувач здійснює HTTP-запит до веб-сервера. Це може бути запит на отримання веб-сторінки, відправлення форми або інша взаємодія з інтерфейсом.

2. Веб-сервер (Веб-сервер -> Flask-застосунок):

- Веб-сервер приймає запит від користувача і передає його до Flask-застосунку для обробки. Flask-застосунок виконує бізнес-логіку, обробляє запит і взаємодіє з іншими компонентами системи.

3. Flask-застосунок (Flask-застосунок -> База даних):

- Якщо обробка запиту потребує даних з бази даних, Flask-застосунок надсилає запит до бази даних. Це може бути запит на вибірку даних, їх зміну або видалення.

4. База даних (База даних -> Flask-застосунок):

- База даних обробляє запит і повертає результат Flask-застосунку.

5. Flask-застосунок (Flask-застосунок -> Google сервер):

- В деяких випадках Flask-застосунок може взаємодіяти із зовнішніми сервісами, такими як Google сервери. Це може бути необхідно для отримання додаткових даних або виконання сторонніх API-запитів.

6. Flask-застосунок (Flask-застосунок -> Веб-сервер):

- Після обробки запиту і збору всіх необхідних даних, Flask-застосунок повертає відповідь веб-серверу.

7. Веб-сервер (Веб-сервер -> Користувач):

- Веб-сервер отримує відповідь від Flask-застосунку і надсилає її користувачу. Це може бути HTML-сторінка, JSON-дані або інша форма відповіді.

Таким чином, ця схема описує потік даних від моменту, коли користувач здійснює запит, до моменту, коли він отримує відповідь, зокрема включаючи обробку запитів на стороні серверу, взаємодію з базою даних та зовнішніми сервісами.

3.2 HTTP маршрутизація в застосунку

Маршрути та їх функції

1. /chat

- Функція: chat

- Опис: Цей маршрут обробляє повідомлення користувача, надіслані у форматі JSON. Він викликає функцію `process_input` для генерації відповіді за допомогою генеративної моделі gemini AI. Відповідь бота надсилається у вигляді JSON-відповіді.

- Методи: POST

2. /

- Функція: index

- Опис: Це головний маршрут застосунку. Він отримує всі жанри та книги з бази даних та відображає головну сторінку. Також перевіряє, чи користувач увійшов у систему, щоб налаштувати вигляд сторінки.

- Методи: GET

3. /login

- Функція: login

- Опис: Цей маршрут обробляє вхід користувача. Він отримує дані для входу та пароля у форматі JSON, перевіряє облікові дані у базі даних і запускає сесію для користувача у разі успішної авторизації.

- Методи: POST

4. /registration

- Функція: registration

- Опис: Цей маршрут обробляє реєстрацію користувача. При GET-запиті він відображає сторінку реєстрації. При POST-запиті обробляє дані реєстрації, додає нового користувача до бази даних і надсилає відповідь.

- Методи: GET, POST

5. /logout (GET)

- Функція: logout

- Опис: Цей маршрут виконує вихід поточного користувача, очищаючи його сесію, і перенаправляє його на головну сторінку.

- Методи: GET

6. /order

- Функція: order

- Опис: Цей маршрут обробляє замовлення книг. Він перевіряє, чи користувач авторизований, і додає вказану книгу до кошика користувача у сесії.

- Методи: POST

7. /deletebook

- Функція: delete_book

- Опис: Цей маршрут видаляє книгу з кошика користувача у сесії. Він перевіряє, чи користувач авторизований, і видаляє вказану книгу з сесії.

- Методи: POST

8. /basket

- Функція: basket

- Опис: Цей маршрут відображає сторінку кошика, показуючи всі книги, що наразі знаходяться у кошику користувача у сесії, та загальну вартість.

- Методи: GET

9. /confirm

- Функція: confirm

- Опис: Цей маршрут підтверджує замовлення користувача. Він обчислює загальну вартість книг у кошику, створює нове замовлення у базі даних і очищає кошик у сесії.

- Методи: POST

10. /book/<int:index>

- Функція: book_detail

- Опис: Цей маршрут відображає деталі конкретної книги, ідентифікованої за її індексом. Якщо книгу знайдено у базі даних, він відображає сторінку з деталями книги, інакше повертає повідомлення про помилку.

- Методи: GET

3.3 Порядок роботи застосунку

1. Головна сторінка (index.html)

- Заголовок: Містить навігаційні посилання, включаючи вхід, вихід, реєстрацію та кошик.

- Список книг: Відображає список усіх книг, отриманих з бази даних (лістинг 3.1).

- Інформація про користувача: Показує інформацію про користувача, якщо він увійшов у систему.

- Поля для вводу данних від акаунта

- Асистент готовий до надання відповідей на запитання користувача, може допомогти з навігацією на сайті, надавати посилання на книги, надавати чітку інформацію про книги, допомагати користувачеві з вибором та навігацією (рис. 3.1).

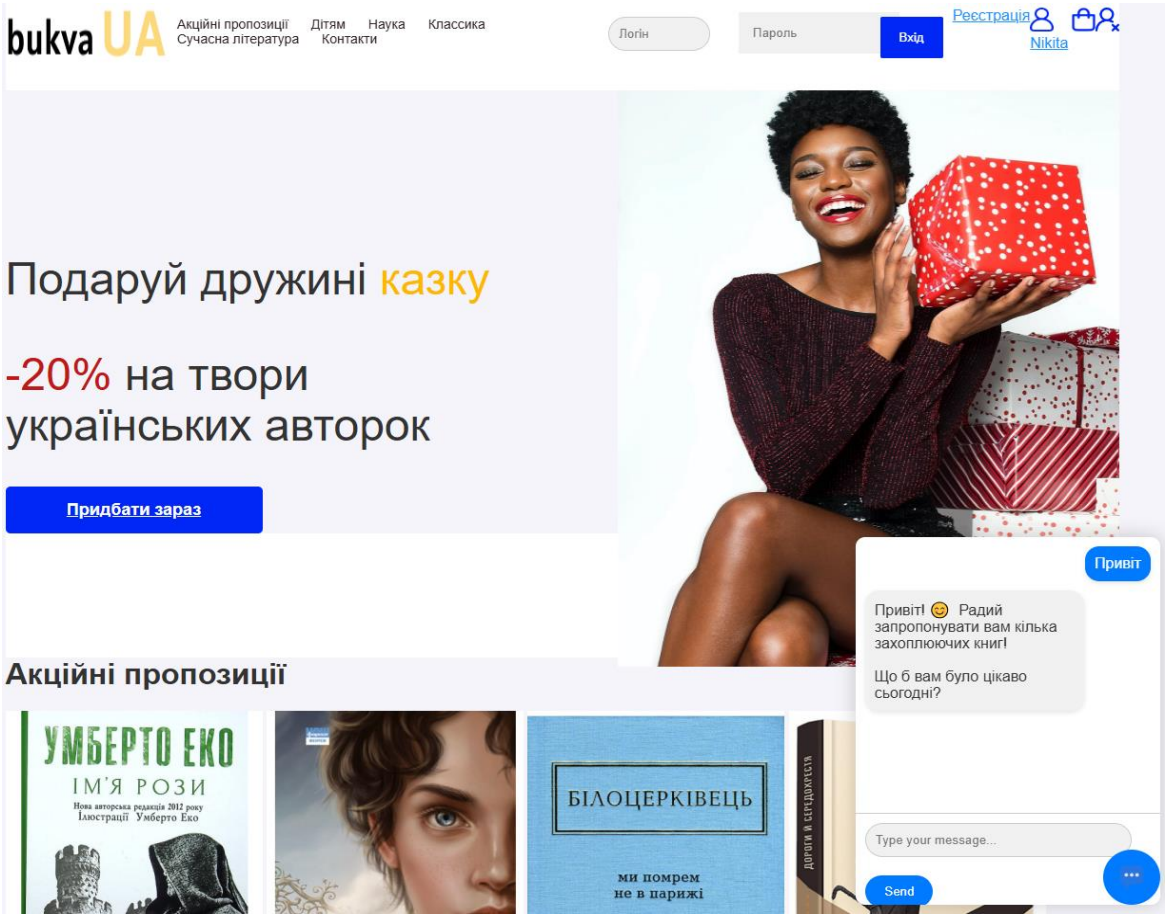

Рис. 3.2. Головна сторінка

Лістинг 3.1. Отримання списку книг з бази даних

```
def index(self):
    if self.db:
        genres = genre.query.all()
        books = book.query.all()
        if 'user' in self.session:
            user = self.session['user']
        else:
            user = None
        return render_template('index.html', genres=genres,
books=books, user=user)
    else:
        message = "Can't connect database"
        return render_template('error.html', message=message)
```

2. Сторінка входу (login.html)

- Форма входу: Містить поля для введення логіна та пароля користувача (рис. 3.3). Здійснення авторизації показано у лістингу 3.2.

- Асистент може допомогти з процесом входу, надати поради щодо введення коректних даних.

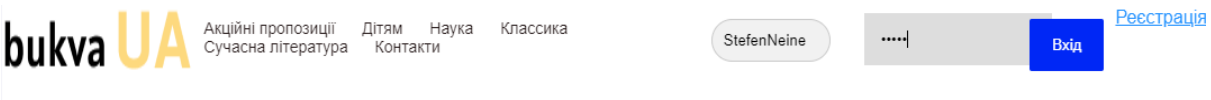

Рис. 3.3. Сторінка входу

Лістинг 3.2. Авторизація користувача

```
def login(self):
    data = request.json
    login = data['login']
    password = data['password']
    if not login or not password:
        return jsonify({"status": "error", "message": "Empty login or
password"}), 400
    if self.db:
        user = customer.query.filter_by(login=login,
password=sha256(password.encode()).hexdigest()).first()
        if user:
```

```
                    self.session["books"] = {}
                    self.session["user"] = user
                    return jsonify({"status": "success", "message":
"Authentication complete successfully", "user": user}), 200
                else:
                    return jsonify({"message": "Authentication failed"}), 401
            else:
                return jsonify({"status": "error", "message": "Database
connection error"}), 500
```

3. Сторінка реєстрації (registrationpage.html)

- Форма реєстрації: Містить поля для введення імені, логіна, пароля та номера телефону користувача (обробка даних показана у лістингу 3.3).

-Асистент може допомогти з заповненням форми реєстрації, надати інформацію про необхідні поля та перевірити коректність введених даних (рис. 3.4).

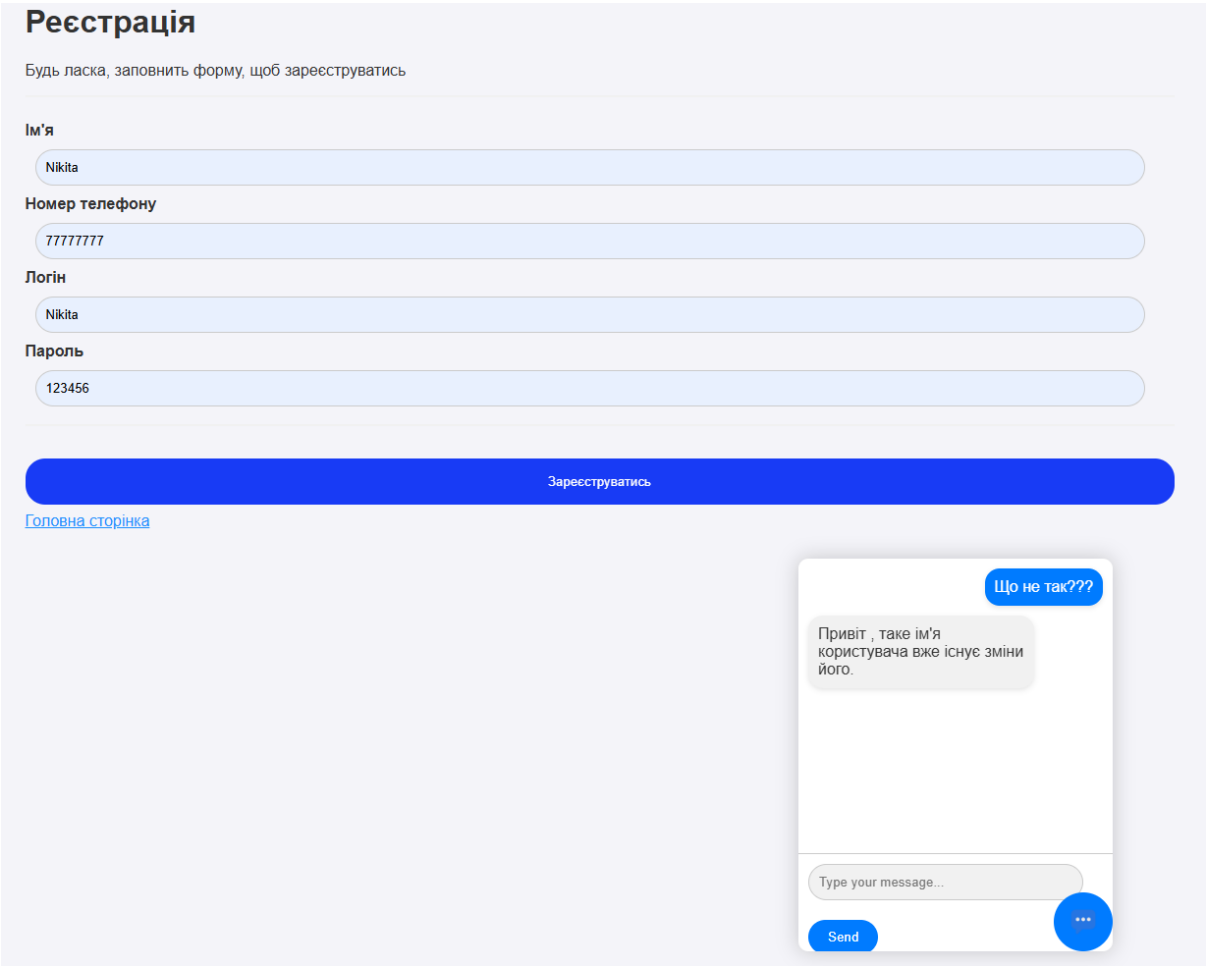

Рис. 3.4 Сторінка реєстрації

Лістинг 3.3. Обробка реєстраційних даних

```
def registration(self):
    if request.method == "POST":
        data = request.json
        addname = data['name']
        addlogin = data['login']
        addpassword = data['password']
        addphone = data['phone']
        if self.db:
            if customer.query.filter_by(login=addlogin).count():
                return jsonify({"status": "error", "message": "User
already exist"}), 422

            user = customer(addname, addphone, addlogin,
sha256(addpassword.encode()).hexdigest())
            self.db.session.add(user)
            self.db.session.commit()
            self.db.session.refresh(user)
            return jsonify({"status": "success", "message":
"Registration successful", "user": user }), 200
        else:
            return jsonify({"status": "error", "message": "Registration
failed"}), 500
    else:
        return render_template('registrationpage.html')
```

4. Сторінка кошика (basketpage.html)

- Список книг: Відображає всі книги, що наразі знаходяться у кошику користувача, з кількостями та цінами (процес показано у лістингу 3.4).

- Загальна вартість: Показує загальну вартість книг у кошику.

- Кнопка підтвердження: Дозволяє користувачеві підтвердити своє замовлення.

- Асистент може надати детальну інформацію про книги в кошику, допомогти з розрахунком загальної вартості та надати інструкції щодо замовлення (рис.3.5).

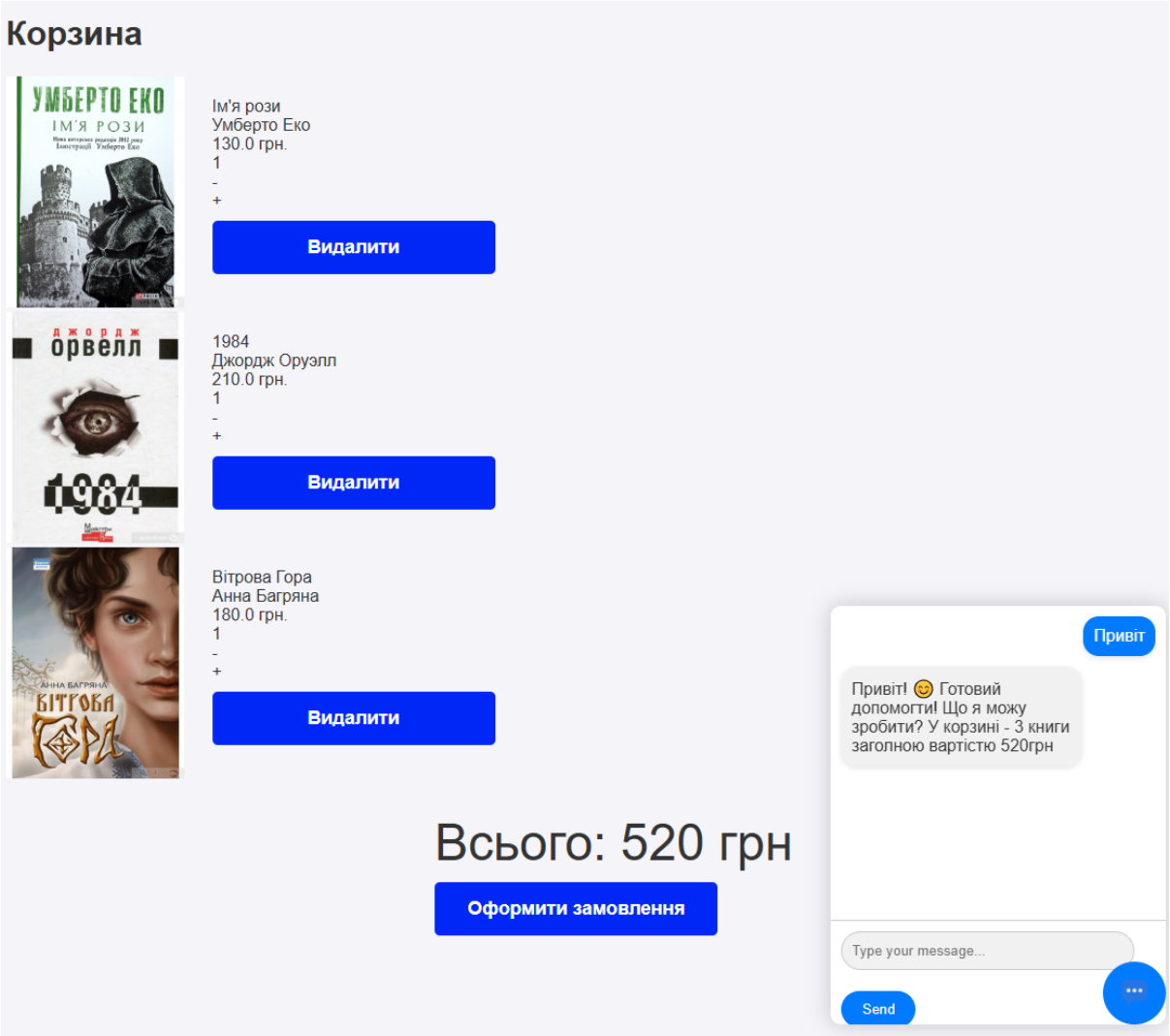

Рис. 3.5. Кошик

Лістинг 3.4. Відображення вмісту кошика

```
def basket(self):
    if 'user' not in self.session:
        return redirect("/")
    total = 0
    for book_id in self.session['books']:
        total += self.session["books"][book_id]["price"] *
self.session["books"][book_id]["quantity"]
    response = render_template('basketpage.html', books =
self.session['books'], user = self.session['user'], total=total)
    return response
```

5. Сторінка деталей книги (book.html)

- Інформація про книгу: Відображає детальну інформацію про обрану книгу, включаючи назву, автора, ціну та зображення (відповідний програмний код подано у лістингу 3.5).

- Асистент може надати додаткову інформацію про книгу, відповісти на запитання щодо автора чи змісту книги, а також запропонувати схожі книги для вибору (рис. 3.6).

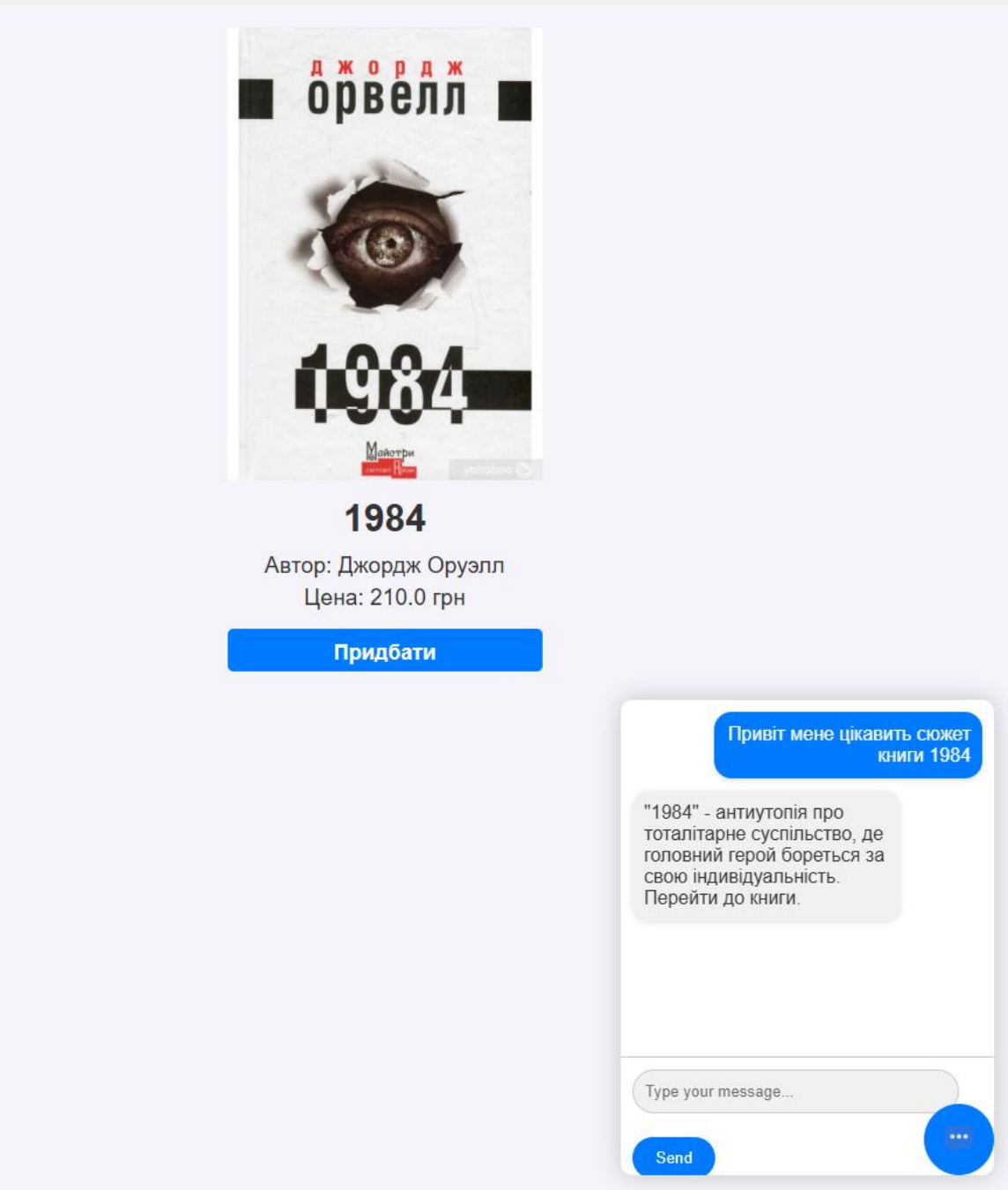

Рис. 3.6. Деталі книги

Лістинг 3.5. Відображення інформації про книгу

```
def book_detail(self, index):
    if self.db:
        _book = book.query.get(index)
        if _book:
            return render_template('book.html', book=_book)
        else:
            return jsonify({"status": "error", "message": "Book not
found"}), 404
    else:
        return jsonify({"status": "error", "message": "Database
connection error"}), 500
```

6. Повідомлення про помилку (error.html)

- Повідомлення про помилку: Відображає повідомлення про помилку у разі виникнення проблем, таких, як проблема з підключенням до бази даних, або якщо не знайдено книгу (рис. 3.7). Процес обробки помилки показаний на лістингу 3.6.

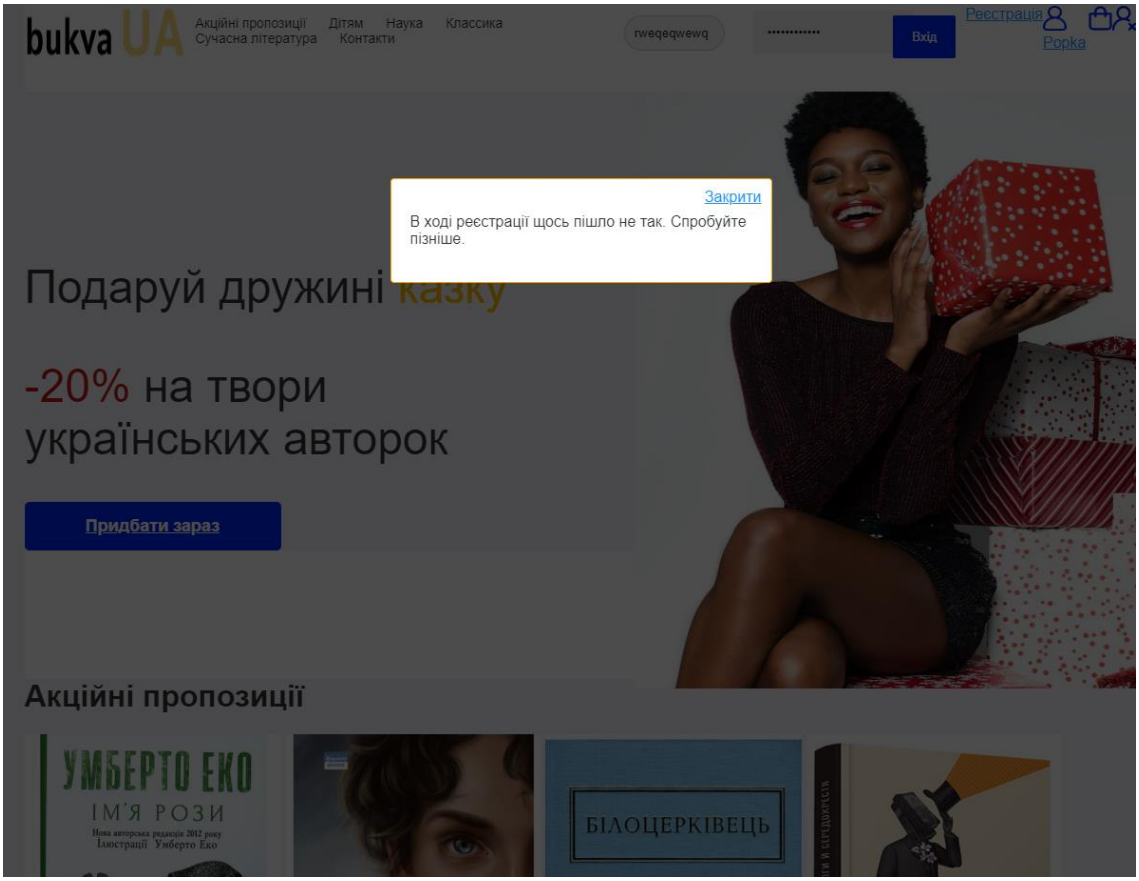

Рис. 3.7. Повідомлення про помилку

Лістинг 3.6. Обробка помилки

```
def index(self):
    if self.db:
        genres = genre.query.all()
        books = book.query.all()
        if 'user' in self.session:
            user = self.session['user']
        else:
            user = None
        return render_template('index.html', genres=genres,
books=books, user=user)
    else:
        message = "Can't connect database"
        return render_template('error.html', message=message)
```

7. Чат з онлайн-ассистентом(`index.html`)

- Асистент може відповідати на запитання клієнта, надавати інформацію про наявність книг або про їх детальний зміст, допомагати користувачеві орієнтуватися на сторінці, пропонувати додаткові книги (рис. 3.8).

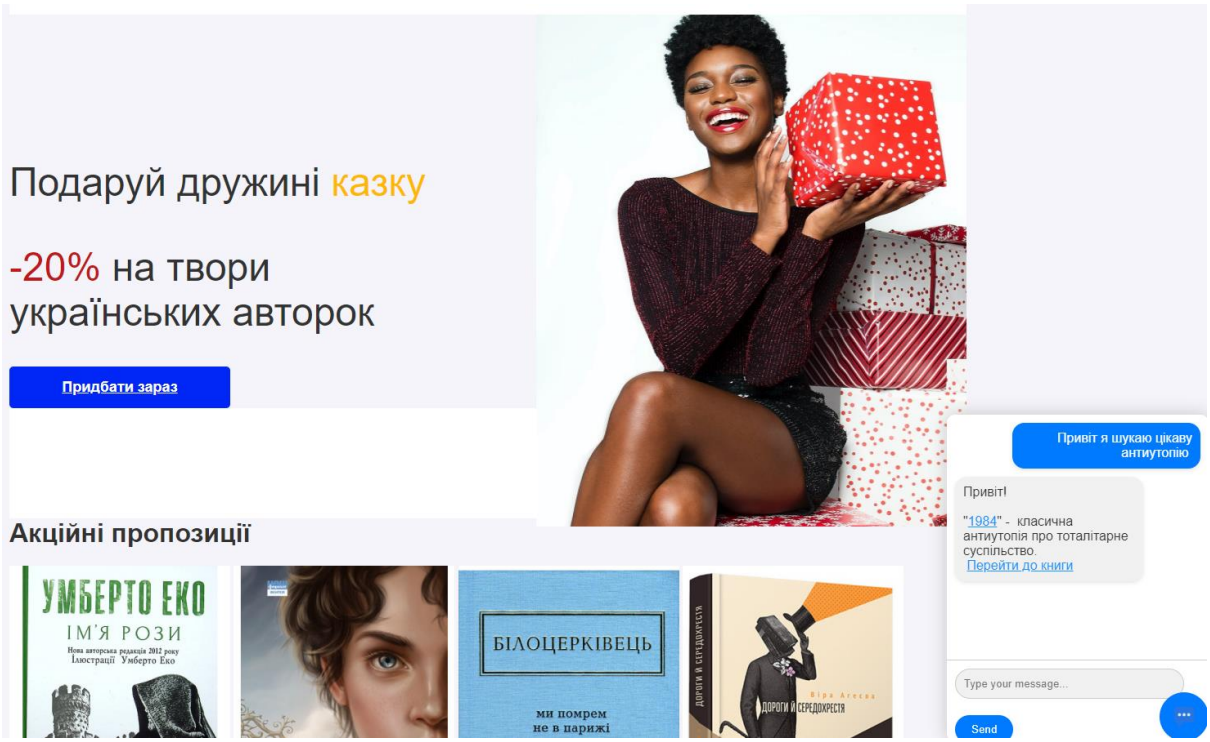

Рис. 3.8. Онлайн-ассистент

Така побудова інтернет-магазину забезпечує легку навігацію по його сторінках та зручний і приємний досвід користувачів.

3.4 Висновки до розділу 3

В ході виконання третьої частини кваліфікаційної роботи був розроблений застосунок, що реалізує інтелектуального чат-бота для книжкового інтернет-магазину «BukvaUA». Було наведено структуру вебзастосунка, у який інтегрується програмний компонент інтелектуального помічника, та детально розписано маршрутизацію запитів сайту.

Також було розроблено опис графічного інтерфейсу програми. Було створено керівництво користувача, яке дозволяє користувачам ознайомитися з функціональністю інтелектуального помічника та отримати обізнаність щодо його використання та взаємодії з ним. Керівництво надає детальні вказівки щодо використання можливостей бота та його участі у різних сценаріях поведінки користувача сайту.

Розроблений застосунок був протестований безпосередньо виконавцем кваліфікаційної роботи вручну.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи «Розробка інтелектуального помічника з використанням хмарних сервісів» було успішно реалізовано ряд завдань та досягнуто мету роботи.

Застосунок написаний мовою Python з використанням фреймворку Flask та СУБД MySQL. У моделі штучного інтелекту була обрана Gemini API AI та хмарний сервіс Google.

Розроблений застосунок дозволяє:

- отримання інтелектуальної поради щодо добору книжок;
- отримання книги щодо фабули книги;
- отримання відповіді щодо сюжету книги у розгорнутому вигляді;
- можливість включення посилань на книжки у контекст відповіді онлайн помічника.

Було проведено тестування функціоналу застосунку. Результати тестування були успішними, що підтверджує його стабільну та ефективну роботу.

Документація, включаючи опис функціоналу застосунку та графічного інтерфейсу користувача, була ретельно підготовлена. Це допоможе користувачам швидко ознайомитися з програмним компонентом, та ефективно використовувати його.

Загальним висновком є те, що розроблений інтелектуальний помічник відповідає поставленим вимогам та має потенціал для подальшого розвитку. У проєкті застосовані сучасні технології програмування і відповідні методи та технології створення та інтеграціїї інтелектуальних чат-ботів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Krupansky J. What Is an Intelligent Digital Assistant? 2017. [Електронний ресурс] – Режим доступу: https://jackkrupansky.medium.com/what-is-an-intelligent-digital-assistant-3f601a4bb1f2

2. Intelligent Virtual Assistant (IVA) [Електронний ресурс] – Режим доступу: https://www.ttec.com/glossary/intelligent-virtual-assistant

3. Kress K. From Bots to Intelligent Virtual Assistants: Building a Digital Worker Factory. [Електронний ресурс] – Режим доступу: https://www.ttec.com/articles/bots-intelligent-virtual-assistants-building-digital-worker

4. Митрошина Н. 15 кращих книжкових інтернет-магазинів України. 2023. [Електронний ресурс] – Режим доступу: https://torgsoft.ua/articles/stati/internet-magazyny-knyg/

5. Yakaboo [Електронний ресурс] – Режим доступу: https://www.yakaboo.ua/

6. Книгарня «Є» [Електронний ресурс] – Режим доступу: https://book-ye.com.ua/

7. «Клуб Сімейного Дозвілля» [Електронний ресурс] – Режим доступу: https://bookclub.ua/

8. Bookovka [Електронний ресурс] – Режим доступу: https://www.bookovka.ua/

9. Book24 [Електронний ресурс] – Режим доступу: https://book24.ua/

10. Balka-book [Електронний ресурс] – Режим доступу: https://balka-book.com/

11. Лавка Бабуїн [Електронний ресурс] – Режим доступу: https://lavkababuin.com/ua/

12. Booklya [Електронний ресурс] – Режим доступу: https://www.booklya.ua/

13. Flask [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Flask

14. Top Free Chatbot tools, APIs, and Open Source models [Електронний ресурс] – Режим доступу: https://www.edenai.co/post/top-free-chatbot-tools-apis-and-open-source-models

15. Gemini AI [Електронний ресурс] – Режим доступу: https://gemini.google.com/

16. OpenAI [Електронний ресурс] – Режим доступу: https://openai.com/

17. Dialog Flow [Електронний ресурс] – Режим доступу: https://cloud.google.com/dialogflow?hl

18. spaCy [Електронний ресурс] – Режим доступу: https://spacy.io/

19. MySQL [Електронний ресурс] – Режим доступу: https://www.mysql.com/

20. Function calling [Електронний ресурс] – Режим доступу: https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling

21. Unhelkar B. Software Engineering with UML. – Auerbach Publications, 2018. – 427 p.

22. Lutz M. Learning Python. O'Reilly Media, 2021. 1200 p.

23. Васильєв О. Програмування в PYTHON. Теорія і практика. Ліра-К, 2023.

24. Nichter D. Efficient MySQL Performance: Best Practices and Techniques. 1st Ed. O'Reilly Media, 2021. 275 p.

25. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення: ДСТУ 3008-95. – Чинний від 1996–01–01. – К.: Держстандарт України, 1996. – 39 с.

26. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги: ДСТУ 3582-97. – Чинний від 1998–07–01. – К.: Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

ДОДАТОК А. ЛІСТИНГ

App.py

```
import os
from flask import *
from flask_sqlalchemy import SQLAlchemy
from flask_wtf.csrf import CSRFProtect

app = Flask (__name__)
app.config ['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:1234@localhost/mydatabase'
app.config ['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:1234@localhost/mydatabase'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
app.secret_key = "An4cFh3d12K656cDpF"



#@app.before_request
#def check_csrf():
#    print("!!!!!!!!!!!!")
#    csrf.protect()
```

Book.py

```
from app import *
from genre import *

class book(db.Model):
    id = db.Column('id', db.Integer, primary_key = True)
    book_name = db.Column(db.String(100))
    book_author = db.Column(db.String(100))
    price = db.Column(db.Float)
    image = db.Column(db.String(200))
    genre_book = db.Column(db.Integer, db.ForeignKey('genre.id'), nullable=False)

    def __init__(self, book_name, book_author, price, image, genre_book):
        self.book_name = book_name
        self.book_author = book_author
```

```
        self.price = price
        self.image = image
        self.genre_book = genre_book

if __name__ == '__main__':
    db.create_all()

    db.session.add(book('Ім`я рози', 'Умберто Еко', 130,   'images/img17.png', 1))
    db.session.add(book('Вітрова гора', 'Анна Багряна', 180  ,'images/img18.png', 1))
    db.session.add(book('Ми помремо не в Парижі','Наталка Білоцерківець', 150, 'images/img19.png',
1))
    db.session.add(book('Дороги і середохрестя', 'Віра Агеєва', 230, 'images/img20.png', 1))
    db.session.add(book('Маленький принс', 'Антуан Сент-Екзопері', 230, 'images/book1.png', 2))
    db.session.add(book('Попелюшка', 'Шарль Перо', 500, 'images/img2.png', 2))
    db.session.add(book('Мій дідусь був черешнею', 'Анджела Нанетті', 230, 'images/img3.png', 2))
    db.session.add(book('Енн із Енворі', 'Люсі-Мод монгомері', 210, 'images/img5.png', 2))
    db.session.add(book('Викрадачі вогню', 'Стівен Котлен', 410, 'images/img6.png', 3))
    db.session.add(book('Момент еврики', 'Джон Кункута', 210, 'images/img7.png', 3))
    db.session.add(book('Фізика', 'Павло Віктор', 300,  'images/img8.png', 3))
    db.session.add(book('Пташина історія', 'Наталія Атамась', 500, 'images/img10.png', 3))
    db.session.add(book('1984', 'Джордж Орвелл', 210,  'images/img13.png', 4))
    db.session.add(book('Село не люди. Добити свідка', 'Люко Дашвар', 90, 'images/img11.png', 4))
    db.session.add(book('Бог завжди подорожує інкогніто', 'Лоран Гунель', 50,  'images/img12.png', 4))
    db.session.add(book('Місто дівчат', 'Елізабет гілберт', 210,'images/img13.png', 4))
    db.session.commit()
    print(book.query.all())
```

books_order.py

```
from app import *
from book import *
from customer import *
from order import *

class books_order(db.Model):
    id = db.Column( db.Integer, primary_key = True)
    quantity = db.Column(db.Integer)
    order_id = db.Column(db.Integer, db.ForeignKey('order.id'), nullable=False)
```

```
    book_id = db.Column(db.Integer, db.ForeignKey('book.id'), nullable=False)

    def __init__(self, quantity, order_id, book_id):
        self.quantity = quantity
        self.book_id = book_id
        self.order_id = order_id


if __name__ == '__main__':
    db.create_all()
#    db.session.add(books_order(1, 1, 1))
    db.session.commit()
#    print(books_order.query.all())
```

Main.py

```
from shop import *

if __name__ == "__main__":
    print("12345")
    book_shop = Shop(__name__)
    book_shop.run(host='0.0.0.0', port=15000)
```

order.py

```
from app import *
from book import *
from customer import *
import datetime
from dataclasses import dataclass

@dataclass
class order(db.Model):
    id: int
    date: str
    amount: str
```

```
    id = db.Column( db.Integer, primary_key = True)
    date = db.Column(db.DateTime)
    amount = db.Column(db.Float)
    user_id = db.Column(db.Integer, db.ForeignKey('customer.id'), nullable=False)

    def __init__(self, amount, user_id):
        self.date = datetime.datetime.now()
        self.amount = amount
        self.user_id = user_id

if __name__ == '__main__':
    db.create_all()
#    user = customer.query.filter_by(login='tester').first()
#    db.session.add(order(100, user.id))
    db.session.commit()
 #    print(order.query.all())
```

Shop.py

```
from flask import *
from flask_wtf.csrf import *
import copy
from flask_sqlalchemy import SQLAlchemy
from app import *
from book import *
from genre import *
from customer import *
from order import *
from books_order import *
from hashlib import sha256
from flask import Flask, render_template, request, jsonify
import os
import vertexai
from vertexai.preview.generative_models import GenerativeModel

def process_input(base_prompt, question):
    full_prompt = f"{base_prompt}, {question}"
    response = model.generate_content(full_prompt)
```

```
    return response.text

@app.route('/chat', methods=['POST'])
def chat():
    user_message = request.json.get('message')
    if user_message:
        bot_response = process_input(base_prompt, user_message)
        global user_response_last, last_response
        user_response_last = user_message
        last_response = bot_response
        return jsonify({'response': bot_response})
    return jsonify({'response': 'No message received'})

class Shop:

    def __init__(self, name):
        self.app = app
        self.db = db
        self.app.add_url_rule("/", "index", self.index)
        self.app.add_url_rule("/login", "login", self.login, methods=['POST'])
        self.app.add_url_rule("/registration", "registration", self.registration, methods=['GET', 'POST'])
        self.app.add_url_rule("/logout", "logout", self.logout)
        self.app.add_url_rule("/order", "order", self.order, methods=['POST'])
        self.app.add_url_rule("/deletebook", "deletebook", self.delete_book, methods=['POST'])
        self.app.add_url_rule("/basket", "basket", self.basket)
        self.app.add_url_rule("/confirm", "confirm", self.confirm, methods=['POST'])
        self.app.add_url_rule("/book/<int:index>", "book_detail", self.book_detail)  # New route for book
detail
        self.session = session

    def book_detail(self, index):
        if self.db:
            _book = book.query.get(index)
            if _book:
                return render_template('book.html', book=_book)
            else:
                return jsonify({"status": "error", "message": "Book not found"}), 404
        else:
```

```
            return jsonify({"status": "error", "message": "Database connection error"}), 500

    def index(self):
        if self.db:
            genres = genre.query.all()
            books = book.query.all()
            if 'user' in self.session:
                user = self.session['user']
            else:
                user = None
            return render_template('index.html', genres=genres, books=books, user=user)
        else:
            message = "Can't connect database"
            return render_template('error.html', message=message)

    def login(self):
        data = request.json
        login = data['login']
        password = data['password']
        if not login or not password:
            return jsonify({"status": "error", "message": "Empty login or password"}), 400
        if self.db:
            user = customer.query.filter_by(login=login,
password=sha256(password.encode()).hexdigest()).first()
            if user:
                self.session["books"] = {}
                self.session["user"] = user
                return jsonify({"status": "success", "message": "Authentication complete successfully", "user":
user}), 200
            else:
                return jsonify({"message": "Authentication failed"}), 401
        else:
            return jsonify({"status": "error", "message": "Database connection error"}), 500

    def registration(self):
        if request.method == "POST":
            data = request.json
            addname = data['name']
```

```
        addlogin = data['login']
        addpassword = data['password']
        addphone = data['phone']
        if self.db:
            if customer.query.filter_by(login=addlogin).count():
                return jsonify({"status": "error", "message": "User already exist"}), 422

            user = customer(addname, addphone, addlogin, sha256(addpassword.encode()).hexdigest())
            self.db.session.add(user)
            self.db.session.commit()
            self.db.session.refresh(user)
            return jsonify({"status": "success", "message": "Registration successful", "user": user }), 200
        else:
            return jsonify({"status": "error", "message": "Registration failed"}), 500
    else:
        return render_template('registrationpage.html')

def logout(self):
    if 'user' in self.session:
        self.session.pop('user')
    if 'books' in self.session:
        self.session.pop("books")
    return redirect(url_for('index'))

def order(self):
    if 'user' not in self.session:
        return jsonify({"status": "error", "message": "Authentication required"}), 401
    data = request.json
    book_id = data['bookid']
    if not book_id:
        return jsonify({"status": "error", "message": "Ordering of unknown book"}), 400
    if self.db:
        books = self.session.pop('books')
        _book = book.query.filter_by(id=book_id).first()
        if _book:
            if book_id in books:
                books[book_id]["quantity"] += 1
            else:
```

```
                books[book_id] = { "book_name": _book.book_name,
                                   "book_author": _book.book_author,
                                   "price": _book.price,
                                  "image": _book.image,"image": _book.image,
                                   "quantity": 1}
            self.session['books'] = books
            return jsonify({"status": "success", "message": "Book ordered successfully"}), 200
        else:
            return jsonify({"status": "error", "message": "Unknown book"}), 400
    else:
        return jsonify({"status": "error", "message": "Database connection error"}), 500

def delete_book(self):
    if 'user' not in self.session:
        return jsonify({"status": "error", "message": "Authentication required"}), 401
    data = request.json
    book_id = data['bookid']
    if not book_id:
        return jsonify({"status": "error", "message": "Ordering of unknown book"}), 400
    try:
        books = self.session.pop('books')
        books.pop(book_id)
        self.session['books'] = books
        return jsonify({"status": "success", "message": "Book deleted"}), 200
    except:
        return jsonify({"status": "error", "message": "Book was not deleted"}), 422

def basket(self):
    if 'user' not in self.session:
        return redirect("/")
    total = 0
    for book_id in self.session['books']:
        total += self.session["books"][book_id]["price"] * self.session["books"][book_id]["quantity"]
    response = render_template('basketpage.html', books = self.session['books'], user =
self.session['user'], total=total)
    return response
def confirm(self):
    if len(self.session["books"]) == 0:
```

```
            return jsonify({"status": "error", "message": "Order is empty"}), 422
        try:
            total = 0
            for book_id in self.session['books']:
                total += self.session["books"][book_id]["price"] * self.session["books"][book_id]["quantity"]
            new_order = order(total, self.session['user']['id'])
            db.session.add(new_order)
            db.session.commit()
            db.session.refresh(new_order)
            for book_id in self.session['books']:
                db.session.add(books_order(self.session["books"][book_id]["quantity"], new_order.id,
book_id))
            db.session.commit()
            self.session["books"] = {}
            return jsonify({ "status": "success", "message": "Successfully confirmed", "order": new_order }),
200
#jsonify({"status: "success", "message": "Order was successfully confirmed"}) #, "order": new_order}),
200
        except Exception as e:
            return jsonify({"status": "error", "message": "Error while configming order"}), 500

    @property
    def get_app(self):
        return self.__app

    def set_app(self, value):
        self.__app = value

    @property
    def get_csrf(self):
        return self.__csrf

    def set_csrf(self, value):
        self.__csrf = value

    @property
    def get_db(self):
        return self.__db
```

```
    def set_db(self, value):
        self.__db = value

    def run(self, **kwargs):
        self.app.run(**kwargs)

os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "asystent-bffo-4dc60211ff96.json"
project_id = 'asystent-bffo'
location = 'us-central1'
vertexai.init(project=project_id, location=location)
model = GenerativeModel('gemini-1.5-pro-preview-0514')
books_file_path = 'books.txt'
books_list = "1:'Ім`я рози', ' Умберто Еко ', 130.00,2:'Вітрова гора', 'Анна Багряна', 180.00,3:'Ми
помремо не в Парижі', 'Наталка Білоцерківець', 150.00,4:'Дороги і середохрестя', 'Віра Агеєва',
230.00,5:'Маленький принс', 'Антуан Сент-Екзопері', 230.00,6:,'Попелюшка', 'Шарль Перо',
500.00,7:'Мій дідусь був черешнею', 'Анджела Нанетті', 230.00,8:'Енн із Енворі', 'Люсі-Мод
Монгомері', 210.00,9:'Викрадачі вогню', 'Стівен Котлен', 410.00,10:'Момент еврики', 'Джон
Кункута', 210.00,11:'Фізика', 'Павло Віктор', 300.00,12:'Пташина історія', 'Наталія Атамась',
500.00,13:'1984', 'Джордж Орвелл', 210.00,14:'Село не люди. Добити свідка', 'Люко Дашвар',
90.00,15:'Бог завжди подорожує інкогніто', 'Лоран Гунель', 50.00,16:'Місто дівчат', 'Елізабет
гілберт', 210.00,"
base_prompt = (f"Ти Олексій онлайн продавець книжок. БУДЬ лаконічний. НЕ радь книжок яких
немає у списку {books_list}. "
               f"після опису книги пиши - Перейти до книгиX де X це номер книги"
               f"Пропоную кілька варінтів книжок"
               f"Коли радишь книжку напиши коротко її тему.")
```

asystent-bffo.json

```
{
  "type": "service_account",
  "project_id": "asystent-bffo",
  "private_key_id": "4dc60211ff96cb50e5acbcef35755556cb823eca",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEuwIBADANBgkqhkiG9w0BAQEFAASCBKUwggShAgEAAoIBAQDaR884X8mdOl8b\nt7LI5b
2ZsrAeFi+NuVwDzz4cHi3rGVNmiWj1b9blN0FArq+DGHEZBRcUGRS8IcA+\n9CjXy0Qtqh/BXhe1D
VeP/5B+zBHXXNJqIF1nOTKNdLmAv788NPA5feJfdrluRH5O\n//X0Edg+YRtiM0leFTQGZxFPUZI7
```

OyOgr3k4G31XrBqUCSliWa9nla8iEB2iEDhA\nGYvQ3Lgzos8kVnmHASe7v+OWOY4KIReBLHMY
SnNIAeDpdVHh1h5QzDmyvSG/Tjlw\nf2uVSRVxirmM2FBL8DRJt4flO6eBxvyMEK7TJNDrklbel/FsH
OnLEXbx2Zyre1L1\n+N6+ZcwtAgMBAAECgf9A1Nxk/Xv/0l+kG+JitrU1HwtCdq/f8Vu1WD0RLy6rH
BqN\n4wtSthLLXTlo4VTkbVTg/hjl1h+ALvPPLKNimTzmwPMJBPKFpjDSiwxWnMX48UaN\nvthIRz
3qRcocM2m0uCkPM4I2oZXNM50HNTXO46KEUkD9l3Tgzw/2+277BKPhBYej\nsJoTuDwPawhAix9
z4GF/oUI/dg8YUc4iux2kvRMIyNVGiu/YeO9AxGn6MHyOCgId\nlenmR5T0k4kW2UG1WhBoN7UPZ
O/hK3dOJBVOwKPtuEH9i1VVejoru8oh1ZeqeZF+\n7CGYHYUDmZKKYbPgYRwCeWLVcHHkgQjk
Sjk7ZVECgYEA+JPaOGIXSRg5N2Hr6Tn7\n41loWIfBe4XwCrvtk8JE+P9ZZyHZaXdB18IdA8uINv7i
BhZ8vgbbdRHD/U0DspM/\n+GxeL6pN+WmDM3JC6sfKninpo+7m4PkrUIbwo6/OsZKvbD0zTGhLkB
vyQmhyYwNF\n4WYVAzI4sotE8FTrtviVSFECgYEA4MxdKJwtuKgHMrSQoAFo0aWHQZxVDiP9Z
wYI\nu4QUqyqpU5tBjEoCqvZufJvPZw5JURs/5uY3YfHkN7SkZzOG6+eR2QAJMzJxPWO5\nF0f5h91
xj8yK+p2Yy3QHWvsrwOfFXqAuXLXFwFC54bn/s8cyObUl7T2hqOsdvUiK\nbB4nKx0CgYAEqlG4K
S8isg+w2GejEvGFTNJGhxW8icD9kpVc5pKA0doNs3bvIcvc\njsoYniNKVPJNBvMEZgCMa+7vvBhLq
7PRp8cEwf+ApNhPiN90pJdK7KJz7zHqGwMT\nGdU+XThV4NjmASgLURHJ/JxJXsh2z0Lqfg3652Bl
CXQbxpw97OBwYQKBgQCvTU88\nkwZooMf0N7Esxcya6I2gkawWeeRrid4XdPm1PXCMXV9xamnI
LqKck9jaUd7E3A4q\nHs+DluHUuw8V163Ym142Lfv+q9B5AiJd0Gf8FqkvbV/X7jwwNIxj28kUBun4T
8u3\nypyqy4kZSTXpWWaGcAhiiJRfqPIpIL+pbp421QKBgFH3+ESF64DyOitg991UffiR\n1rWlHZfQx
vRRZ11UTnIJ/6CgrUPZLpg8pwsdQHJwtIleXX6ZWfGvMnxI8ZLam5BL\n1+8hty7pgBl9jpCiFTbhF8p
7hVhuSTioQxuVMe9zNUszjjDDyAb4aa5YkeivydTr\nWqYNQcVmUfSoZHKKUT8x\n-----END
PRIVATE KEY-----\n",
  "client_email": "asistance@asystent-bffo.iam.gserviceaccount.com",
  "client_id": "105930899575119776733",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/asistance%40asystent-
bffo.iam.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}

Mydatabase.sql

```
-- Create database if not exists
CREATE DATABASE IF NOT EXISTS mydatabase;
USE mydatabase;

-- Drop tables if they exist
DROP TABLE IF EXISTS books_order;
```

```
DROP TABLE IF EXISTS book;
DROP TABLE IF EXISTS genre;
DROP TABLE IF EXISTS customer;

-- Create customer table
CREATE TABLE customer (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    phone BIGINT,
    login VARCHAR(255),
    password VARCHAR(255)
);

-- Create genre table
CREATE TABLE genre (
    id INT AUTO_INCREMENT PRIMARY KEY,
    genre_name VARCHAR(255)
);

-- Create book table
CREATE TABLE book (
    id INT AUTO_INCREMENT PRIMARY KEY,
    book_name VARCHAR(255),
    book_author VARCHAR(255),
    price DECIMAL(10, 2),
    genre_book INT NOT NULL,
    image VARCHAR(255),
    FOREIGN KEY (genre_book) REFERENCES genre(id)
);

-- Create books_order table
CREATE TABLE books_order (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    book_id INT NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customer(id),
    FOREIGN KEY (book_id) REFERENCES book(id)
);
```

```
-- Insert data into customer table
INSERT INTO customer (name, phone, login, password) VALUES
('skorpion', 993548567, 'skorpion', 'skorpion'),
('Valencia', 80995638596, 'nodoubts', 'nodebts'),
('Emispir', 80664659145, 'ripsime', 'nalbandyan'),
('Рус', 80994758935, 'Rus', 'Ruslan'),
('Настенька', 80507563986, 'vasha', 'mymra');

-- Insert data into genre table
INSERT INTO genre (genre_name) VALUES
('Акційні пропозиції'),
('Дітям'),
('Наука'),
('Классика'),
('Сучасна література');

-- Insert data into book table
INSERT INTO book (book_name, book_author, price, genre_book, image) VALUES

('Ім`я рози', 'Умберто Еко', 130.00, 1, 'images/img1.png'),
('Вітрова гора', 'Анна Багряна', 180.00, 1, 'images/img2.png'),
('Ми помремо не в Парижі', 'Наталка Білоцерківець', 150.00, 1, 'images/img3.png'),
('Дороги і середохрестя', 'Віра Агеєва', 230.00, 1, 'images/img4.png'),
('Маленький принц', 'Антуан Сент-Екзоперi', 230.00, 2, 'images/img5.png'),
('Попелюшка', 'Шарль Перо', 500.00, 2, 'images/img6.png'),
('Мій дідусь був черешнею', 'Анджела Нанетті', 230.00, 2, 'images/img7.png'),
('Енн із Енворі', 'Люсі-Мод Монгомері', 210.00, 2, 'images/img8.png'),
('Викрадачі вогню', 'Стівен Котлен', 410.00, 3, 'images/img9.png'),
('Момент еврики', 'Джон Кункута', 210.00, 3, 'images/img10.png'),
('Фізика', 'Павло Віктор', 300.00, 3, 'images/img11.png'),
('Пташина історія', 'Наталія Атамась', 500.00, 3, 'images/img12.png'),
('1984', 'Джордж Орвелл', 210.00, 4, 'images/img13.png'),
('Село не люди. Добити свідка', 'Люко Дашвар', 90.00, 4, 'images/img14.png'),
('Бог завжди подорожує інкогніто', 'Лоран Гунель', 50.00, 4, 'images/img15.png'),
('Місто дівчат', 'Елізабет гілберт', 210.00, 4, 'images/img16.png');
```