ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка веб-додатку для створення тематичних повідомлень з використанням моделей генеративного штучного інтелекту»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки

освітньо-професійної програми Штучний інтелект

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Артем КЛІПЕНШТЕЙН
(підпис)

Виконав: здобувач вищої освіти група ШІД-41
Артем КЛІПЕНШТЕЙН

Керівник:
д.т.н., професор Євген ЧИЧКАРЬОВ

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кліпенштейна Артема Андрійовича

1. Тема кваліфікаційної роботи: Розробка веб-додатку для створення тематичних повідомлень з використанням моделей генеративного штучного інтелекту

керівник кваліфікаційної роботи Євген ЧИЧКАРЬОВ д.т.н., професор,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024р.

3. Вихідні дані до кваліфікаційної роботи:
   3.1 Науково-технічна література з питань, пов'язаних із застосуванням аналізу даних;
   3.2 Практичний досвід аналізу даних;
   3.3 Концепція побудови веб-додатків.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   4.1 Дослідження принципів побудови веб-додатку;
   4.2 Дослідження принципів інтегрування штучного інтелекту;

4.3 Аналіз технологій створення веб-додатку та можливості його застосування.

5. Перелік графічного матеріалу: *презентація*
   5.1. Тема дипломної роботи;
   5.2. Мета роботи. Об’єкт дослідження. Предмет дослідження;
   5.3. Результат дослідження фреймворків для веб-розробки;
   5.4. Результат дослідження та опис реалізації програми;
   5.5. Апробація результатів дослідження;
   5.6. Висновки.

6. Дата видачі завдання «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	27.02-08.03.24	Виконано
2	Вивчення матеріалів для аналізу розвитку штучного інтелекту	08.03-18.03.24	Виконано
3	Дослідження технологій створення веб-додатків	18.03-27.03.24	Виконано
4	Дослідження технологій створення чат-бота-консультанта	27.03-10.04.24	Виконано
5	Розробка веб-додатку	10.04-21.04.24	Виконано
6	Інтеграція штучного інтелекту	21.04-02.05.24	Виконано
7	Оформлення роботи: вступ, висновки, реферат	02.05-18.05.24	Виконано
8	Розробка демонстраційних матеріалів	18.05-31.05.24	Виконано

Здобувач вищої освіти _____ Артем Кліпенштейн
(підпис)

Керівник
кваліфікаційної роботи _____ Євген Чичкарьов
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 53 стор., 2 табл., 20 рис., 18 джерел.

Мета дослідження – підвищення ефективності комунікацій між компаніями та клієнтами завдяки впровадженню чат-бота-консультанта на основі штучного інтелекту. З'ясувати можливості наявних технологій для реалізації веб-сайту з інтегрованим чат-ботом-консультантом на основі штучного інтелекту.

Об'єкт дослідження – процес підвищення ефективності комунікацій між компаніями та клієнтами завдяки впровадженню чат-бота-консультанта на основі штучного інтелекту.

Предмет дослідження – сукупність технологій створення веб-сайту та інтеграції чат-бота-помічника на основі штучного інтелекту.

Короткий зміст роботи: У роботі проведено огляд існуючих веб-додатків з інтегрованим штучним інтелектом. Проведено огляд можливих застосувань штучного інтелекту у веб-додатках. Проведено аналіз існуючих технологій для створення веб-додатків. Проведено аналіз існуючих технологій для інтеграції штучного інтелекту.

Створено та проаналізовано діаграми класів, діаграми послідовностей та діаграми прецендентів.

Для реалізації веб-додатку було використано Flask, HTML, CSS, JavaScript, SQLite та AssistantAPI.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, ЧАТ-БОТ, ВЕБ-ДОДАТОК, CSS, HTML. JAVASCRIPT, FLASK, ASSISTANTAPI

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня бакалавра

Направляється здобувач Кліпенштейн А.А. до захисту кваліфікаційної роботи

за спеціальністю 122 Комп'ютерні науки

освітньої програми Штучний інтелект

на тему: «Розробка веб-додатку для створення тематичних повідомлень з використанням моделей генеративного штучного інтелекту».

Кваліфікаційна робота і рецензія додаються.

Директор ННІ ІТ _____ Андрій БОНДАРЧУК
(підпис)

Висновок керівника кваліфікаційної роботи

Здобувач Кліпенштейн А.А. під час виконання кваліфікаційної роботи показав відмінну теоретичну та практичну підготовку. Проявив самостійність, ініціативність, високу старанність, добрі знання теоретичного матеріалу по спеціальним дисциплінам, креативність при рішенні типових інженерних рішень та вміння правильно використовувати науково-технічну літературу.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Кліпенштейна Артема Андрійовича на оцінку «відмінно» та присвоїти йому кваліфікацію бакалавр з комп'ютерних наук.

Керівник кваліфікаційної роботи _____ Євген Чичкарьов
(підпис)

"___"_____ 2024 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Кліпенштейн А.А. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою Штучного інтелекту _____ Ольга ЗІНЧЕНКО
(підпис)

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну роботу
на здобуття освітнього ступеня бакалавра

здобувача вищої освіти Кліпенштейна Артема Андрійовича
на тему: «Розробка веб-додатку для створення тематичних повідомлень з використанням моделей генеративного штучного інтелекту».

Актуальність:

З розвитком технологій генеративного штучного інтелекту, таких як GPT, можливості створення тематичних повідомлень значно розширилися. Використання цих технологій дозволяє автоматизувати процес підтримки клієнтів за домогою спеціально навчених чат-ботів-консультантів, що є надзвичайно іноваційним для різних сфер, включаючи маркетинг, освіту та бізнес-комунікації.

Тема дипломної роботи є актуальною, оскільки вона спрямована на впровадження інноваційних технологій у сферу веб-розробки, які дозволяють значно підвищити ефективність та продуктивність зворотного зв'язку з клієнтами, зменшуючи витрати часу та людських ресурсів, необхідних для цього.

Позитивні сторони:

1. Кваліфікаційна робота здобувача Кліпенштейна Артема виконана відповідно до поставленого завдання та в повному обсязі.
2. Розробка веб-додатку для створення тематичних повідомлень з використанням моделей генеративного штучного інтелекту надає детальні інструкції та технології для підвищення ефективності комунікацій між компаніями та клієнтами, завдяки впровадженню чат-бота-консультанта, що дозволяє значно зменшити потребу в ручній праці та час, необхідний для цього.
3. Кваліфікаційна робота, його пояснювальна записка, графічний матеріал виконані з дотриманням відповідних вимог.

Недоліки:

1. Для роботи системи потрібні якісні вхідні дані, що може бути проблемою у випадку некоректно сформованих або нерелевантних запитів користувачів.
2. Необхідність регулярного оновлення та навчання моделей для підтримання актуальності та точності генерації тематичних повідомлень, що потребує додаткових ресурсів і часу.

Висновок: кваліфікаційна робота на здобуття ступеня бакалавра заслуговує оцінку "відмінно", а здобувач Кліпенштейн Артем Андрійович заслуговує присвоєння кваліфікації: бакалавр з комп'ютерних наук.

Рецензент: _____ _____
підпис *Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ВСТУП

Актуальність теми. В нинішньому динамічно розвиваючомуся цифровому світі електронна комерція стала фундаментальним елементом глобальної економіки, змінюючи традиційні підходи до взаємодії бізнесу з клієнтами.

З точки зору конкуренції, на ринку, де важливо виділятися серед інших, інтеграція помічників штучного інтелекту на платформах електронної комерції стає дедалі важливішою. Компанії, які застосовують рішення на базі штучного інтелекту, сприймаються як інноваційні та орієнтовані на клієнта, приваблюючи технологічно підкованих споживачів та підвищуючи лояльність до бренду.

Миттєвий зворотний зв'язок та персоналізована взаємодія, що надають помічники на основі штучного інтелекту, значно полегшують ручну працю та час, потрібний для цього. Ці помічники можуть виконувати широкий спектр завдань, від відповідей на поширені запитання, надання загальної інформації о компанії, до порад з покупок, базуючись на історії переглядів та покупок користувача. Це не лише прискорює процес покупок, але й робить його більш привабливим та адаптованим до індивідуальних переваг, таким чином підвищуючи задоволеність і лояльність клієнтів.

Оскільки цифрова комерція продовжує набирати оберти, інтеграція штучного інтелекту – це необхідність для компаній, які прагнуть забезпечити виключне обслуговування клієнтів, підвищити ефективність комунікацій та залишатися конкурентоспроможними.

Мета дослідження – підвищення ефективності комунікацій між компаніями та клієнтами завдяки впровадженню чат-бота-консультанта на основі штучного інтелекту. З'ясувати можливості наявних технологій для реалізації веб-сайту з інтегрованим чат-ботом-консультантом на основі штучного інтелекту.

Об'єкт дослідження – процес підвищення ефективності комунікацій між компаніями та клієнтами завдяки впровадженню чат-бота-консультанта на основі штучного інтелекту.

Предмет дослідження – сукупність технологій створення веб-сайту та інтегрування чат-бота-помічника на основі штучного інтелекту.

Завдання дослідження:

1. Проаналізувати предметну область:

– Оглянути поняття штучного інтелекту;
– Оглянути розумні чат-боти популярних компаній.

2. Оглянути засоби реалізації:

– Провести огляд мови програмування та інших технологій для реалізації візуальної частини веб-додатку;
– Оглянути наявні технології створення чат-бота-консультанта.

3. Спроектувати та реалізувати веб-додаток з використанням чат-бота-консультанта:

– Побудувати діаграму класів;
– Побудувати діаграму послідовностей;
– Побудувати діаграму прецедентів;
– Провести огляд основного механізму роботи;
– Розробити веб-додаток-бібліотеку, з інтегрованим чат-ботом консультантом.

Методи дослідження:

– вивчення існуючих аналогів;
– вивчення існуючих технологій;
– методи об’єктно-орієнтовного моделювання;
– побудова тестових програм;
– відлагодження тестових програм або додатку в цілому.

Теоретична, методична та практична значущість:

Теоретична значущість полягає в розвитку та поглибленні знань про інтеграцію штучного інтелекту в веб-додатки. Дослідження підтверджує важливість використання сучасних технологій штучного інтелекту, таких як чат-боти, для підвищення ефективності комунікації між компаніями та клієнтами. Отримані результати також можуть стати основою для подальших наукових досліджень у галузі веб-розробки та інтеграції штучного інтелекту, сприяючи розвитку нових методів і підходів у цій сфері.

Методична значущість полягає у розробці та апробації методів створення веб-додатку з інтеграцією моделей генеративного штучного інтелекту. Запропоновані методи дозволяють ефективно використовувати наявні технології для розробки веб-додатків, що включають чат-боти та інші елементи штучного інтелекту. Це забезпечує розробникам інструменти для впровадження сучасних технологій у практичну діяльність, сприяючи підвищенню якості та ефективності розробки програмного забезпечення.

Практична значущість отриманих результатів полягає в створенні працюючого веб-додатку, який має усі необхідні функції для взаємодії з користувачами за допомогою чат-бота-консультанта на основі штучного інтелекту. Цей додаток може бути використаний як у комерційних, так і в освітніх цілях, забезпечуючи ефективну комунікацію та підвищуючи задоволеність клієнтів. Розроблені підходи та рішення можуть бути застосовані для подальшого вдосконалення інших веб-додатків, що робить результати дослідження корисними для широкого кола користувачів та розробників.

1 АНАЛІЗ ТЕХНОЛОГІЙ

1.1 Огляд веб-додатків з імлементованим чат-ботом-консультантом

1. H&M

Веб-сайт H&M пропонує широкий асортимент модного одягу, взуття та аксесуарів для жінок, чоловіків та дітей. Користувачі можуть переглядати нові колекції, дізнаватися про останні модні тенденції, купувати товари онлайн та перевіряти наявність товарів у місцевих магазинах. Сайт також містить інформацію про знижки та спеціальні пропозиції.

Чат-бот H&M (рис. 1.1) використовується для покращення клієнтського досвіду, допомоги користувачам у пошуку товарів, надання персоналізованих рекомендацій та відповіді на запитання щодо замовлень і доставок. Це дуже допомагає знижувати навантаження на службу підтримки та забезпечує швидку і ефективну взаємодію з клієнтами.

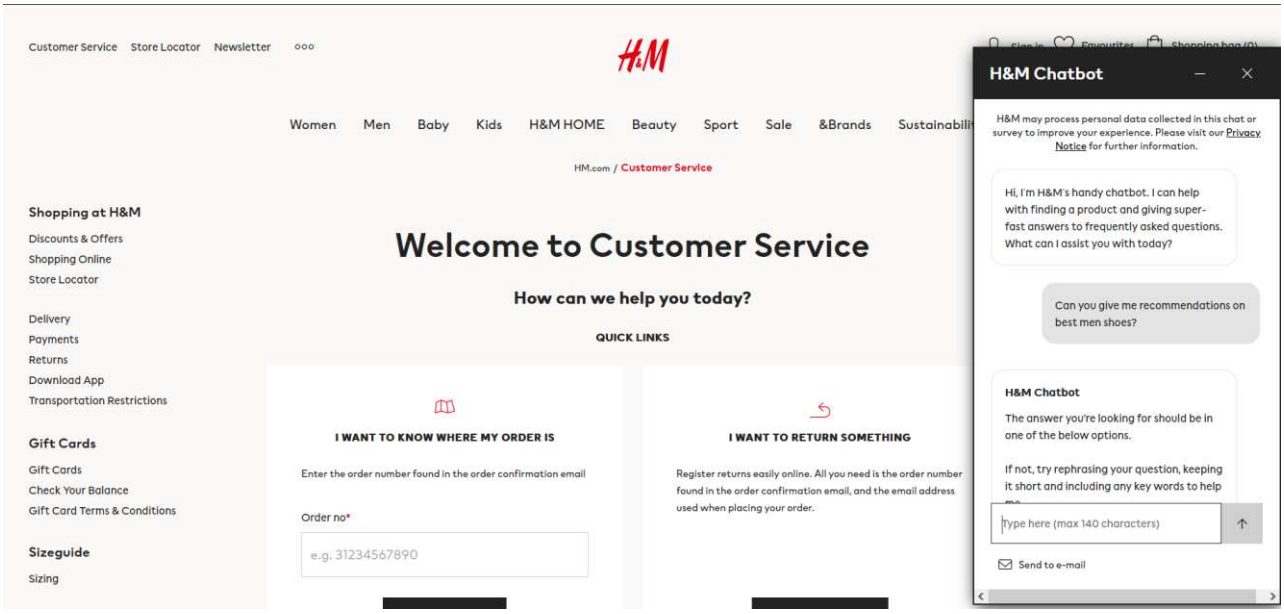


Рис. 1.1 Чат-бот H&M

2. Abu Dhabi Islamic Bank (ADIB)

Веб-сайт ADIB пропонує широкий спектр банківських послуг для індивідуальних та корпоративних клієнтів. Користувачі можуть знайти інформацію про різні фінансові продукти, такі як кредитні картки, особисті

кредити, іпотеку та інші банківські послуги. Сайт також надає можливість керування рахунками онлайн, перегляду транзакцій, оплату рахунків, а також підтримку клієнтів.

Чат-бот ADIB (рис 1.2) використовується для покращення обслуговування клієнтів, надання швидких відповідей на запитання та полегшення доступу до банківських послуг. Допомагає користувачам отримувати інформацію про рахунки та транзакції, консультує з приводу банківських продуктів та послуг, а також забезпечує безпечний спосіб взаємодії з банком.

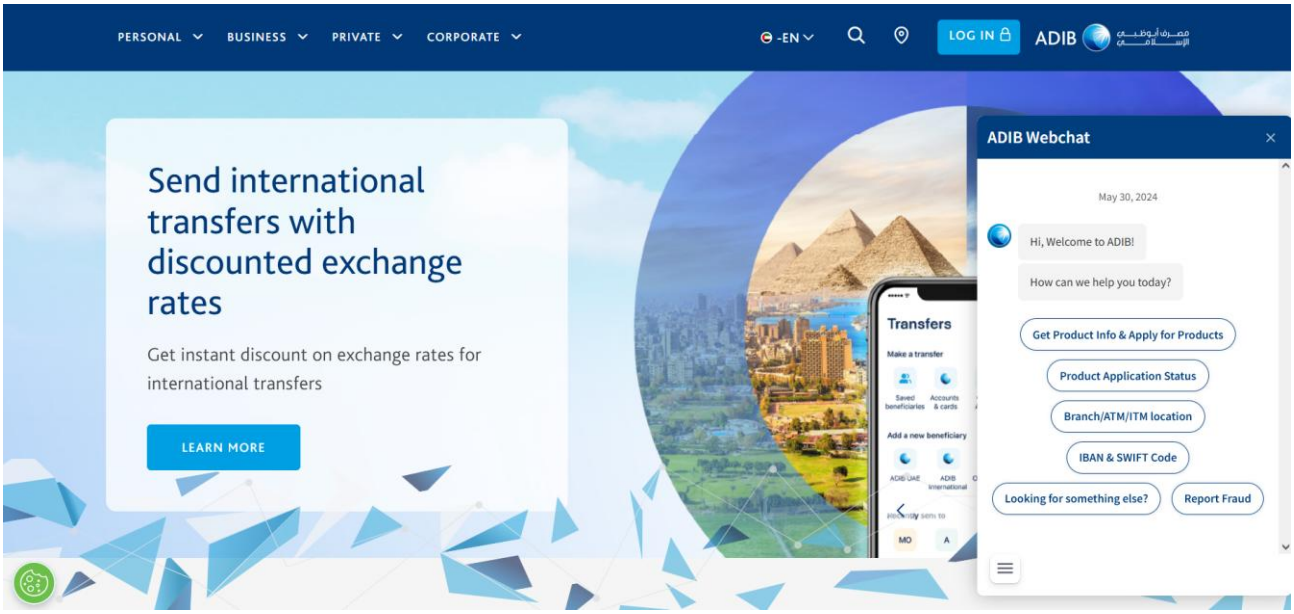


Рис. 1.2 Чат-бот ADIB

3. Sephora

Веб-сайт Sephora спеціалізується на продажу косметики, засобів для догляду за шкірою, парфумерії та аксесуарів. Користувачі можуть переглядати широкий асортимент продуктів, читати відгуки, дивитися навчальні відео та купувати товари онлайн. Сайт також надає інформацію про спеціальні пропозиції, новинки та ексклюзивні продукти.

Чат-бот Sephora (рис. 1.3) використовується для надання користувачам персоналізованих консультацій, рекомендацій щодо косметичних продуктів та можливості віртуальної примірки макіяжу. Він допомагає користувачам знайти потрібні товари, отримати поради з догляду за шкірою та макіяжем, а також надає

інформацію про замовлення та доставку. Це покращує загальний користувацький досвід та залучає клієнтів до взаємодії з брендом.

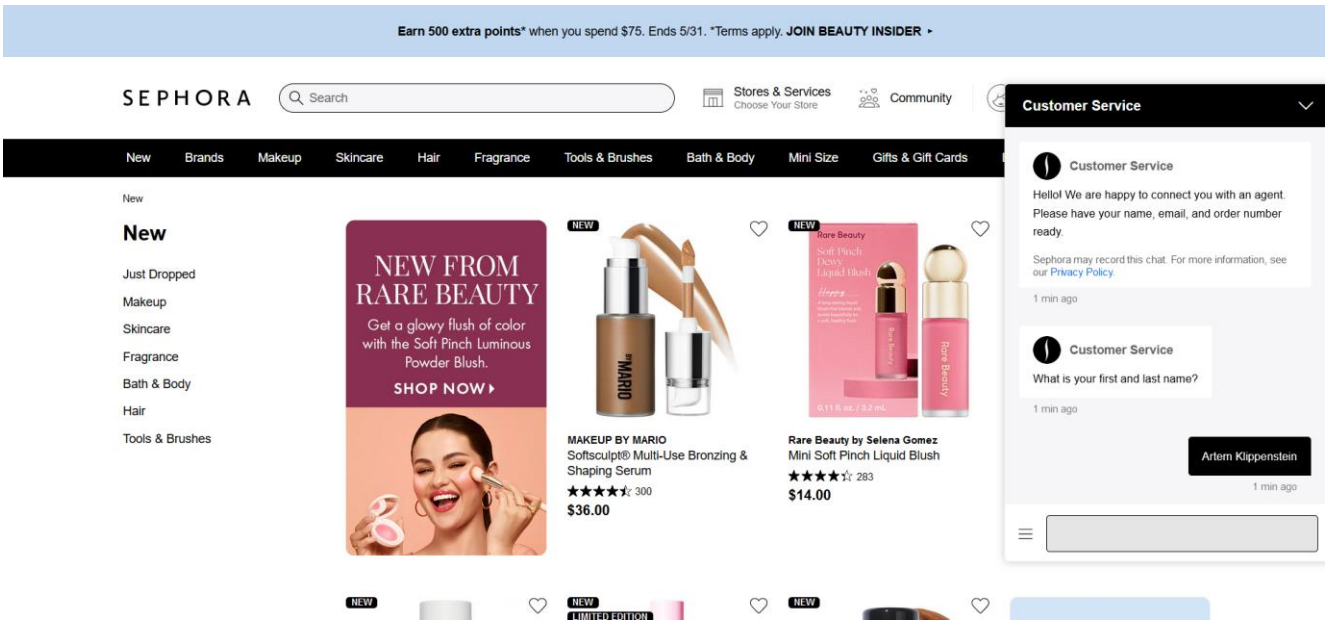


Рис. 1.3 Чат-бот Sephora

4. Peloton

Веб-сайт Peloton пропонує інтерактивні фітнес-продукти та послуги. Peloton спеціалізується на фітнес-обладнанні, яке включає велотренажери, бігові доріжки та інше обладнання з інтегрованими екранами для доступу до онлайн-занять.

Чат-бот Peloton (рис. 1.4) допомагає вирішувати різноманітні питання, від консультацій щодо вибору обладнання до технічної підтримки та допомоги з підписками. Давайте більш детально розглянемо його можливості:

– Надання детальної інформації про фітнес-обладнання Peloton, його характеристики та можливості;
– Консультації щодо процесу покупки, допомога у виборі відповідного обладнання та оформлення замовлення;
– Пояснення, як підписатися на онлайн-заняття, доступ до тренувань та використання фітнес-платформи;
– Відповіді на технічні питання щодо використання обладнання, налаштування, підключення до інтернету та усунення несправностей;
– Відповіді на загальні запитання, наприклад, про доставку, повернення

товарів, умови гарантії та інші питання, пов'язані з обслуговуванням клієнтів.

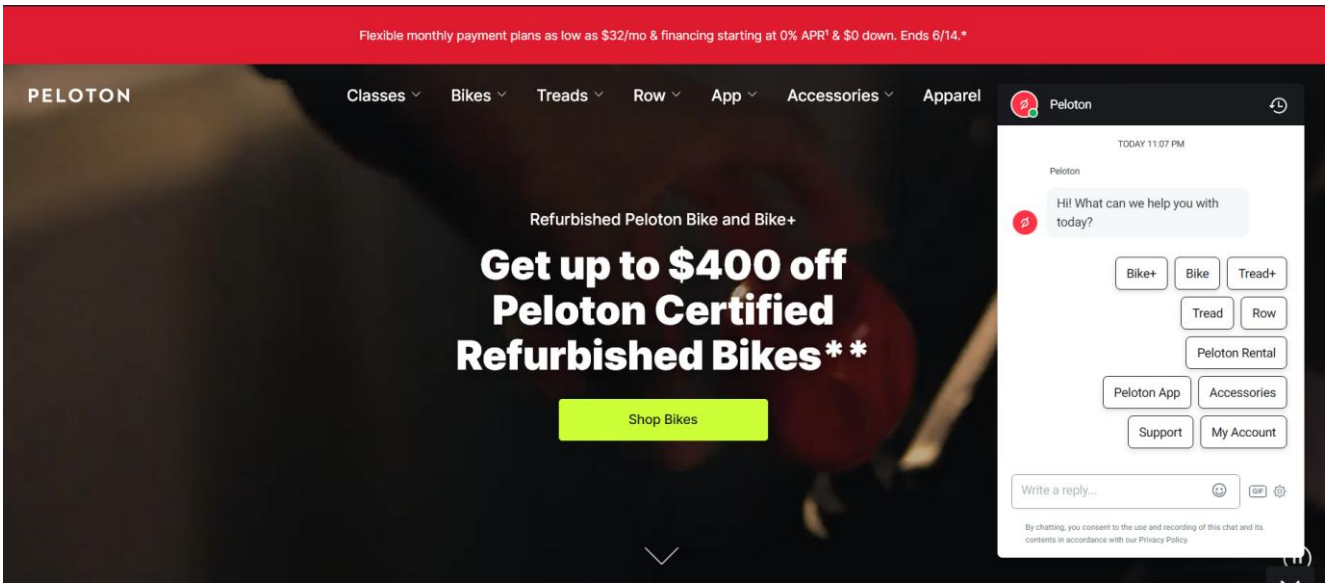


Рис. 1.4 Чат-бот Peloton

1.2 Технології створення сайтів

1.2.1 Веб-фреймворки

«Веб-фреймворки являють собою набір бібліотек та модулів, що дозволяють писати додатки, не переймаючись низькорівневими деталями, такими як протоколи, управління потоками тощо». [1]

Інакше кажучи, веб-фреймворки – це програмне забезпечення, яке пропонує спосіб створення та запуску веб-додатків. Таким чином, вам не потрібно писати код самостійно та шукати можливі помилки та недоліки.

«На ранніх етапах розробки веб-додатків веб-фреймворки були введені як засіб припинення ручного кодування додатків, де лише розробник конкретного додатка міг його змінювати. На сьогодення ми маємо мови, спеціально створені для вебу, і залежно від вашого завдання, ви можете вибрати один веб-фреймворк, який задовольняє всі ваші вимоги, або комбінувати декілька фреймворків.» [2]

«Мегафреймворки (такі як Django) приймають рішення за вас, зменшуючи складність і надаючи готові інструменти для конфігурації, розробки та тестування. Однак, якщо їхній підхід вам не підходить, ви можете зіткнутися з

труднощами через їхні обмеження. З іншого боку, мікрофреймворки (такі як Flask або FastAPI) не нав'язують жодних рішень, що робить їх легкими для початку роботи. Але коли ваш проект масштабується, ви залишаєтесь наодинці». [3]

Давайте детально розглянемо такі веб-фреймворки:

1. Django

«Django – це високорівневий веб-фреймворк на Python, який дозволяє швидко розробляти безпечні та легкосупроводжувані веб-сайти. Розроблений досвідченими програмістами, Django знімає з вас більшу частину клопоту веб-розробки, тож ви можете зосередитися на написанні свого додатка, не винаходячи колесо заново. Він є безкоштовним і з відкритим вихідним кодом, має активну та процвітаючу спільноту, чудову документацію та багато варіантів безкоштовної та платної підтримки.» [4]

Вибір Django як веб-фреймворка має ряд переваг, особливо для початківців і професіоналів. Ось декілька причин, чому ви можете вибрати Django:

- Django дотримується філософії "batteries-included", що означає, що він надає величезну кількість вбудованих функцій та можливостей. Це дозволяє розробникам зосередитися на створенні додатка, а не на налаштуванні та конфігурації інструментів, необхідних для розробки.
- «Django використовує архітектуру Model-Template-View (MTV) – це добре структурований і організований спосіб розробки веб-додатків. MTV розділяє дані (Model), користувацький інтерфейс (Template) та бізнес-логіку (View), що полегшує управління та підтримку коду». [5]
- Django включає потужний ORM (Object-Relational Mapping), який дозволяє розробникам взаємодіяти з базою даних за допомогою об'єктів Python замість написання SQL-запитів. Це спрощує операції з базою даних і знижує ризик SQL-ін'єкцій.

«Проаналізувавши веб-фреймворк Django, можна стверджувати, що він є найпопулярнішим для веб-розробки, написаним мовою програмування Python і використовує архітектуру MVC (Model-Template-View). Django ідеально підходить для проєктів з обмеженим часом і невеликим бюджетом. Якщо додаток

не є дуже об'ємним і потрібне лише просте маршрутизування URL і шаблони з простим контекстом, тоді замість Django можна використовувати фреймворк Flask. Фреймворк для створення додатка на Flask у результатуючих HTML-файлах містить додаткові теги та скрипти. Механізм роботи обробки форм також працює інакше. Flask зберігає більше інформації про поля форми на сторінках. Розмір HTML-файлів, створених за допомогою фреймворку Flask, виявляється значно більшим». [6]

2. Flask

«Flask – це веб-фреймворк для Python, що надає розробникам мінімальний набір інструментів для розробки веб-додатків із можливістю вибору бібліотек та інструментів. Він спроектований так, щоб бути легким і гнучким, що спрощує його використання для створення простих веб-додатків і служить міцною основою для більш складних програм шляхом підключення будь-яких необхідних компонентів у міру необхідності». [7]

Flask надає основи веб-розробки, такі як маршрутизація та обробка запитів, але залишає все інше на розсуд розробника, дозволяючи здійснювати широке налаштування за допомогою доступних розширень.

Flask базується на інструментарії WSGI (інтерфейс шлюзу веб-сервера) та шаблонному двигуні Jinja2. WSGI є стандартом для розробки веб-додатків на Python. Його вважають специфікацією універсального інтерфейсу між веб-сервером та веб-додатком. Jinja2 – це шаблонний двигун для вебу, який поєднує шаблон із певним джерелом даних для рендерингу динамічних веб-сторінок.

3. FastAPI

«FastAPI – це високопродуктивний веб-фреймворк з відкритим вихідним кодом для створення API з використанням Python починаючи з версії 3.6. Запущений у 2018 році, він був створений Себастіаном Рамірезом, який не був задоволений існуючими фреймворками, такими як Flask і DRF. Тому він створив свій власний фреймворк, використовуючи інструменти, такі як Starlette та Pydantic. Сьогодні багато всесвітньо відомих компаній, таких як Uber, Netflix та Microsoft, використовують FastAPI для створення своїх додатків». [8]

Його перша перевага - високий рівень продуктивності, порівнянний з NodeJS і Go завдяки Starlette та pydantic. Цей фреймворк також дуже швидкий у написанні коду, що прискорює розробку.

Кількість багів і можливостей для людських помилок значно зменшена.

FastAPI дуже інтуїтивно зрозумілий у використанні, з функціями автозавершення та відлагодження.

FastAPI розроблений бути простим у вивченні та використанні, цей інструмент зменшує час, витрачений на читання документації. Дублювання коду також зведене до мінімуму.

FastAPI забезпечує готовий до виробництва код з автоматичною інтерактивною документацією. Він базується на відкритих стандартах OpenAPI та JSON Schema.

«Три найпоширеніші фреймворки для розробки веб-додатків на Python - це Django, Flask і FastAPI. Кожен з них має свої переваги та недоліки в залежності від конкретного випадку використання». [9]

4. Ruby on Rails

«Ruby on Rails є одним із найпопулярніших фреймворків для розробки веб-додатків на мові Ruby. Заснований на принципах "Convention over Configuration" та "Don't Repeat Yourself", він був створений Девідом Хейномейєром Ганссоном і вперше випущений у 2005 році. Rails популярний серед розробників через свою ефективність у створенні складних веб-додатків за допомогою порівняно невеликої кількості коду» [10].

Ruby on Rails використовує модель-контролер-вид (MVC) архітектуру, що дозволяє розділити додаток на три основні компоненти: модель, яка відповідає за базу даних, контролер, який керує бізнес-логікою, та вид, який є інтерфейсом користувача. Вимагає мінімуму конфігурації, щоб почати роботу, оскільки багато налаштувань вже передбачені за замовчуванням.

Convention over Configuration (CoC) надає перевагу конвенціям перед налаштуваннями. Це означає, що, дотримуючись певних конвенцій при найменуванні і структуруванні коду, розробники можуть уникнути написання

багатьох конфігураційних файлів, спрощуючи тим самим розробку.

DRY Principle (Don't Repeat Yourself) є одним із фундаментальних принципів Rails, який спонукає розробників уникати дублювання коду. Це сприяє створенню більш чистого, ефективнішого коду з легшим супроводом.

1.2.2 Front-end технології

Обираючи фронтенд-технології для створення веб-додатку, спершу потрібно вирішити, чи використовуватимете ви фреймворки, чи писатимете код на класиці – HTML, CSS та JavaScript. Для опанування фундаментальних знань рекомендується спочатку освоїти HTML, CSS та JavaScript.

Фреймворки суттєво спрощують процес написання коду завдяки вбудованим функціям та абстракціям, які автоматизують повторювані завдання та структурують код за допомогою готових шаблонів. Однак, новачки можуть зіткнутися з труднощами, не розуміючи, як ці інструменти працюють «під капотом».

Опанування класики допомагає новачкам краще зрозуміти основи веб-розробки та дізнатися, як браузери інтерпретують код, що є дуже важливим для глибшого розуміння більш складних технологій і підходів. На практиці, це забезпечує міцну основу та здатність вирішувати проблеми, які можуть виникати під час використання фреймворків.

Проте, коли проєкт розростається, а вимоги стають дедалі складнішими, фреймворки можуть значно спростити розробку. Вони вносять у процес структуру та узгодженість, що є важливим для підтримки великих або розподілених команд розробників. Фреймворки також часто містять вбудовані рішення для часто виникаючих проблем, таких як управління станом, маршрутизація, асинхронні запити і безпека, що може суттєво прискорити процес розробки.

Таким чином, важливо зважити потреби проекту та рівень комфорту розробників з основними технологіями. Для невеликих проектів або особистого навчання є більш доцільним обрання класичного тріо, коли для великих та складних проектів, чи коли потрібна швидка розробка, використання фреймворків

буде доречним рішенням. Отже, вибір між класикою і фреймворками залежить від конкретної ситуації і вимог до проекту, а також від того, наскільки глибоко розробник хоче зануритись в деталі реалізації веб-технологій.

Давайте детально розглянемо HTML, CSS і JavaScript, а потім популярні фронтенд фреймворки:

1. HTML, CSS і JavaScript

HTML (HyperText Markup Language), CSS (Cascading Style Sheets), і JavaScript становлять основу фронтенд-розробки, кожен з цих інструментів відіграє свою унікальну роль у створенні веб-сторінок та додатків.

HTML є кістяком будь-якої веб-сторінки. Це мова розмітки, яка використовується для структурування контенту на веб-сторінках. За допомогою HTML ви визначаєте елементи такі як заголовки, абзаци, списки, посилання, зображення, таблиці та багато інших об'єктів, які взаємодіють з користувачем. Важливим аспектом HTML є його семантичні теги, такі як <article>, <section>, <header>, <footer> та <nav>, які допомагають вказати призначення блоків контенту, сприяючи кращій організації та доступності контенту.

«Каскадні таблиці стилів (CSS) – це мова таблиць стилів, яка дозволяє авторам і користувачам додавати стилі до структурованих документів, таких як HTML-сторінки. Каскадні таблиці стилів були розроблені для забезпечення точного контролю за інтервалом між символами, вирівнюванням тексту, розташуванням об'єктів на сторінці, характеристиками шрифтів, кольором, фонами тощо». [11]

«JavaScript – це мова програмування, яка додає інтерактивність і динамічний контент до статичних HTML-сторінок. Більшість динамічних ефектів, таких наведення миші, анімація, випадаюче меню та валідація форм, зазвичай виконуються завдяки JavaScript тим чи іншим чином». [12]

Від простих завдань, таких як перевірка форм перед відправленням на сервер, до складних додатків або ігор. Ця мова може взаємодіяти з HTML та CSS для маніпуляції елементами сторінки у реальному часі. Сучасний JavaScript підтримує модульність, асинхронні функції, та багато інших можливостей, які

роблять його могутнім інструментом для фронтенд та бекенд розробки.

Таким чином, HTML, CSS та JavaScript формують основу, на якій будується більшість сучасних веб-додатків. HTML надає структуру веб-сторінки, CSS визначає її зовнішній вигляд, а JavaScript додає інтерактивність та функціональність. Разом вони дозволяють створювати багаті, динамічні та інтерактивні користувацькі інтерфейси.

2. Фреймворки

1) «Angular – це open-source JavaScript-фреймворк, розроблений і підтримуваний Google, який забезпечує стандартну структуру та додаткові функції для спрощення розробки веб- та мобільних додатків». [13]

Однією з визначальних характеристик Angular є використання звичайного Document Object Model (DOM) у поєднанні з TypeScript (статично типізованим надмножиною JavaScript) для покращення читабельності та підтримуваності коду.

Angular є всебічним фреймворком, ідеальним для розробки великомасштабних корпоративних веб-додатків. Він забезпечує структуроване середовище з вбудованою функціональністю для маршрутизації, керування формами тощо, що допомагає підтримувати послідовність у великих проектах.

2) «Vue – це open-source JavaScript фреймворк, створений Еваном Ю у 2014 році як альтернатива важкішим фреймворкам, таким як AngularJS і React. Vue поєднує підходи, вплив яких видно з Angular, та спрощені функції для фронтенд-інтерфейсу та розробки додатків. Основна бібліотека Vue зосереджена лише на шарі представлення і призначена для поступового впровадження в проекти». [14]

Основна бібліотека Vue легко інтегрується з іншими бібліотеками та існуючими проектами. Vue легко освоюється, головним чином завдяки зосередженню фреймворку лише на шарі представлення бібліотеки. Vue також здатен забезпечувати роботу складних односторінкових додатків завдяки використанню сучасних інструментів та підтримувальних бібліотек для додаткової функціональності.

Окрім цього, Vue використовує методи двостороннього зв’язування даних, для гарантування, що різні частини додатку, які використовують одні й ті ж дані, завжди матимуть однакову версію цих даних (зазвичай останню версію).

Підсумовуючи, Vue забезпечує баланс між надійністю Angular і простотою React. Його легко інтегрувати з іншими проектами та бібліотеками, і він є фантастичним для нових розробників завдяки своїй простоті та зрозумілій документації. Vue найкраще використовувати в односторінкових програмах, а також може обробляти більш складні програми в масштабі.

3) «React – це JavaScript-бібліотека для створення інтерфейсів користувача (UI) на веб-платформі. React є декларативною, компонентною бібліотекою, що дозволяє розробникам створювати багаторазові UI-компоненти.». [15]

React використовує підхід Virtual DOM (віртуальна модель об’єктів документа), який оптимізує продуктивність рендерингу, мінімізуючи оновлення DOM. React працює швидко та добре поєднується з іншими інструментами та бібліотеками.

Однією з ключових привабливих рис використання React є його декларативний підхід – він робить код більш зрозумілим і легшим для відлагодження, оскільки описує, як ваш додаток повинен виглядати в різних станах, замість того, щоб деталізувати покрокові інструкції для досягнення цього вигляду. Це робить ваш код більш передбачуваним і легшим для роботи як для вас, так і для інших членів вашої команди.

4) «Svelte – це open-source фронтенд JavaScript-фреймворк для створення інтерактивних веб-сторінок. Загальна концепція Svelte подібна до вже існуючих фреймворків, таких як React і Vue, оскільки дозволяє розробникам створювати веб-додатки». [16]

Однак, на відміну від React і Vue, Svelte пропонує кілька особливостей, які надають розробникам унікальний досвід, а саме:

1) Написання меншої кількості рядків коду, що економить час, зменшує кількість багів і підвищує читабельність. Svelte намагається забезпечити це, запроваджуючи простий формат, написаний на TypeScript.

2) Відсутність Virtual DOM (віртуальна модель об’єктів документа). Virtual DOM - це спосіб оновлення стану шляхом порівняння знімка попереднього дерева об'єктів з поточним. Virtual DOM використовується в React. Svelte є компілятором, тому користувачеві не потрібно завантажувати бібліотеку в браузер для запуску коду Svelte. Натомість на сторінці завантажується простий .js файл для рендерингу додатку. Всі оновлення об'єктів здійснюються під час компіляції, що допомагає Svelte зменшити накладні витрати, які створюються віртуальним DOM. Крім того, відсутність необхідності завантажувати всю бібліотеку значно зменшує розмір файлу. Це особливо корисно для мобільних пристроїв.

Вибір між Angular, Vue, React і Svelte залежить від вимог проєкту, досвіду команди та масштабу програми. Angular підходить для корпоративних додатків, де потрібна міцна структура. Vue забезпечує баланс між гнучкістю та потужністю, роблячи його придатним для широкого спектра проєктів. React з його багатою екосистемою та компонентною архітектурою ідеальний для великих програм з динамічними інтерфейсами. Інноваційний підхід Svelte підходить для проєктів, де важливі швидкість і невеликий розмір. Розуміння сильних сторін кожного з цих фреймворків дозволяє розробникам обрати найбільш відповідну технологію для своїх потреб.

1.2.3 Database технології

1) «SQLite – це вбудована, серверна система управління реляційними базами даних. Це бібліотека з відкритим вихідним кодом, що працює в оперативній пам’яті, не потребує конфігурації та встановлення. Крім того, вона дуже зручна, оскільки її розмір менший за 500 кілобайт, що значно менше порівняно з іншими системами управління базами даних. добре підходить для додатків, де потрібне вбудовування легкої бази даних у саму програму». [17]

- SQLite використовує динамічні типи для таблиць. Це означає, що ви можете зберігати будь-яке значення в будь-якому стовпці, незалежно від типу даних.
- SQLite дозволяє одному з'єднанню з базою даних одночасно отримувати

доступ до декількох файлів бази даних. Це приносить багато приємних можливостей, таких як об'єднання таблиць у різних базах даних або копіювання даних між базами даних за допомогою однієї команди.

– SQLite здатен створювати бази даних в оперативній пам'яті, які дуже швидкі у роботі.

2) «MySQL – це система управління реляційними базами даних з відкритим вихідним кодом, яка зберігає дані в структурованому форматі за допомогою рядків і стовпців. Це програмне забезпечення, яке дозволяє користувачам створювати, керувати та маніпулювати базами даних. Розроблена компанією MySQL AB, яка зараз належить Oracle Corporation, MySQL відома своєю надійністю, масштабованістю та простотою у використанні». [18]

Реляційна база даних – це цифровий сховок, що збирає дані та організовує їх відповідно до реляційної моделі. У цій моделі таблиці складаються з рядків і стовпців, а відносини між елементами даних підкоряються суворій логічній структурі. RDBMS (Relational database management system) – це набір програмних інструментів, які використовуються для фактичної реалізації, управління та запитів до такої бази даних.

Взаємодія з базою даних MySQL здійснюється за допомогою SQL (Structured Query Language). SQL не є повноцінною мовою програмування. Але як мова запитів, вона пропонує простий синтаксис для керування вашою базою даних через:

– Створення, оновлення та видалення таблиць;
– Індексацію таблиць;
– Отримання, вставку, оновлення та видалення даних у таблицях;
– Об'єднання даних з декількох таблиць;
– Виконання математичних функцій над запитаними даними;
– Розподіл даних.

Продуктивність MySQL є високомасштабованою, щоб підтримувати навіть найбільші додатки. Це можна зробити за допомогою методів оптимізації, таких як індексація таблиць, оновлення обладнання та горизонтальне шардінгування.

3) PostgreSQL – це високопродуктивна система управління реляційними базами даних з відкритим вихідним кодом, яка підтримує як SQL, так і JSON запити. Вона була розроблена спільнотою більше 20 років тому. Завдяки своєй багатофункціоності, PostgreSQL використовується як основне сховище даних або дата-склад для різних веб-, мобільних, геопросторових та аналітичних додатків.

PostgreSQL має потужний набір функцій, таких як простори таблиць, асинхронна реплікація, вкладені транзакції, резервне копіювання в режимі online/hot, а також вдосконалений планувальник/оптимізатор запитів. Вона підтримує міжнародні набори символів, багатобайтові кодування та Unicode, а також більшість типів даних SQL:2008, включаючи INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL та TIMESTAMP.

Система також здатна зберігати великі бінарні об'єкти, такі як зображення, звуки, відео та карти. Вона підтримує зовнішні ключі, об'єднання, подання, тригери та збережені процедури, а також інтеграцію з різними мовами програмування та протоколами, включаючи Python, Java, Perl, .Net, Go, Ruby, C/C++, Tcl та ODBC.

PostgreSQL враховує локальні налаштування для сортування, чутливості до регістру та форматування. Ця база даних є високомасштабованою, здатною обробляти великі обсяги даних і одночасно обслуговувати багато користувачів.

Підсумовуючи, SQLite підходить для простих додатків з мінімальними вимогами до бази даних і незначною масштабованістю. Вона ідеальна для ситуацій, коли база даних більше схожа на файл, ніж на складну реляційну систему. MySQL балансує між простотою використання та можливістю працювати зі складними програмами, особливо веб-додатками. MySQL популярна в проєктах, де продуктивність і масштабованість важливіші за розширені функції SQL. PostgreSQL найкраще підходить для додатків, де потрібна висока надійність, розширені можливості SQL і обробка складних типів даних, що особливо важливо в академічних, наукових, фінансових та інших областях.

1.3 Як використовувати елементи штучного інтелекту у веб-додатках

Варіант 1. Чат-боти

Чат-боти автоматизують відповіді на запити користувачів, допомагають з навігацією по сайту і підтримкою клієнтів. Їх можна реалізувати за допомогою різних платформ:

AssistantAPI від OpenAI: Платформа, що дозволяє створювати чат-ботів на основі потужних мовних моделей, таких як GPT-4. Вона підтримує обробку природної мови, генерацію тексту, діалоги і відповіді на запитання на різних мовах, включаючи українську.

Dialogflow від Google: Інструмент для створення інтуїтивних інтерфейсів розмовного ШІ, який підтримує кілька мов та інтегрується з платформами, такими як Facebook Messenger, Slack, Telegram тощо. Використовується для розпізнавання намірів користувачів і генерації відповідей.

Microsoft Bot Framework: Набір інструментів для розробки чат-ботів, включає SDK, інструменти та сервіси. Підтримує інтеграцію з каналами, такими як Skype, Microsoft Teams, Facebook Messenger, що дозволяє створювати ботів, які відповідають на запитання, виконують завдання та взаємодіють з користувачами.

Варіант 2. Рекомендаційні системи

Рекомендаційні системи пропонують користувачам продукти чи контент на основі їх попередньої активності. Такі системи використовуються, наприклад, Netflix і Amazon. Ось переглянемо технології, що вони використовують:

Collaborative Filtering (фільтрація на основі співпраці): Метод рекомендацій, що використовує схожість між користувачами чи елементами для прогнозування інтересів. Наприклад, якщо двоє користувачів мають схожі уподобання, те, що подобається одному, ймовірно, сподобається й іншому.

Content-Based Filtering (контентна фільтрація): Метод, що ґрунтується на властивостях елементів. Якщо користувачеві сподобався певний елемент, система рекомендує інші, подібні за характеристиками. Наприклад, якщо користувач переглядає фільми одного жанру, йому будуть рекомендуватися фільми цього ж жанру.

Варіант 3. Обробка природної мови (NLP)

SpaCy: Бібліотека для обробки природної мови на Python. Підтримує завдання, такі як токенізація, частковий аналіз мови (POS tagging), іменоване розпізнавання сутностей (NER) тощо.

NLTK (Natural Language Toolkit): Інструментарій для обробки природної мови на Python, що включає різноманітні інструменти та ресурси для роботи з текстами. Використовується для навчальних і дослідницьких цілей.

BERT (Bidirectional Encoder Representations from Transformers): Модель глибокого навчання від Google, що може розуміти контекст. Використовується для різних завдань NLP, таких як класифікація тексту, аналіз настроїв та питання-відповідь.

Варіант 4. Обробка зображень та відео

OpenCV (Open Source Computer Vision Library): Бібліотека для обробки зображень та відео, що включає різноманітні алгоритми для комп'ютерного зору. Використовується для завдань, таких як розпізнавання об'єктів, обробка зображень та відеоаналіз.

YOLO: Алгоритм для розпізнавання об'єктів в реальному часі, що використовується для ідентифікації об'єктів на зображеннях і відео.

DeepFace: Бібліотека для розпізнавання облич, використовується для аутентифікації користувачів, аналізу емоцій та ідентифікації облич.

Варіант 5. Аналіз даних та прогнозування

Scikit-learn: Популярна бібліотека для машинного навчання на Python, що включає інструменти для класифікації, регресії, кластеризації та зменшення вимірів. Використовується для побудови і навчання моделей на основі даних.

TensorFlow: Фреймворк для машинного навчання від Google, який дозволяє створювати і навчати нейронні мережі. Підтримує розробку як простих моделей, так і складних глибоких нейронних мереж для завдань комп'ютерного зору, NLP та інших.

PyTorch: Фреймворк для машинного навчання від Facebook. Популярний серед дослідників та інженерів завдяки своїй динамічній природі і підтримці GPU для прискореного навчання моделей.

2 ЗАСОБИ І ТЕХНОЛОГІЇ СТВОРЕННЯ ВЕБ-ДОДАТКУ

2.1 Вибір технологій розробки веб-додатку

Інтеграція Flask з HTML, CSS та JavaScript створює цілісне середовище для розробки веб-додатків. Flask відповідає за серверну логіку та взаємодію з базою даних, а HTML створює структуру веб-сторінок. CSS визначає стиль та візуальне оформлення цих сторінок, а JavaScript забезпечує динамічність і інтерактивність.

Давайте детально розглянемо кожну з перелічених технологій:

2.1.1 Flask

Flask є ідеальним вибором для мого проєкту через його простоту та гнучкість. Оскільки він мінімалістичний і модульний, Flask дозволяє розпочати з основних функцій і додавати потрібні розширення лише тоді, коли це необхідно. Завдяки інтеграції з Python, я можу використовувати численні бібліотеки для роботи з базами даних, обробки даних та інших завдань.

Flask, будучи мікрофреймворком для веб-розробки на Python, відомий своєю легкістю у використанні та гнучкістю. На відміну від більш складних фреймворків, таких як Django, Flask надає лише необхідні інструменти для створення веб-додатків, дозволяючи розробникам обирати інструменти на власний розсуд. Це робить його ідеальним для малих та середніх проєктів, де потрібен контроль над усіма компонентами додатка.

Flask служить ядром програми, обробляючи маршрутизацію URL, запити і відповіді, а також виступаючи посередником між інтерфейсом і базою даних. Він направляє HTTP-запити до відповідних Python-функцій, які обробляють ці запити і повертають відповіді у вигляді HTML-сторінок, JSON або інших форматів. Flask підтримує різні розширення, які додають такі функції, як валідація форм, автентифікація користувачів і управління базами даних.

Flask може відображати HTML-файли або динамічно генерувати HTML-контент за допомогою шаблонів Jinja2, вбудовуючи змінні і керуючі структури в HTML-код. Він взаємодіє з базами даних, такими як SQLite, для зберігання, отримання, оновлення і видалення даних, забезпечуючи динамічне управління

контентом залежно від взаємодії користувачів або змін у даних.

2.1.2 SQLite

Я обрав SQLite через його простоту і легкість налаштування. Оскільки він не вимагає налаштування окремого серверного процесу, SQLite чудово підходить для розробки та невеликих виробничих додатків. Простота і легкість SQLite дозволяє налаштувати швидку роботу програми без додаткових витрат на складні системи управління базами даних.

SQLite надає компактну дискову базу даних, яка не потребує окремого серверного процесу і дозволяє отримувати доступ за допомогою стандартного SQL. Він широко використовується в додатках, які потребують базових операцій з даними, але не потребують масштабованості повноцінної системи управління базами даних.

SQLite інтегрується безпосередньо у Flask-додаток. Всі дані, таблиці, індекси і структура зберігаються в єдиному файлі, який може бути використаний на різних платформах. SQLite пропонує надійний механізм транзакцій без необхідності додаткової конфігурації. Flask взаємодіє з SQLite через SQL-запити, які виконуються в коді Python, а отримані дані можуть бути передані у фронтенд для динамічного відображення змін на веб-сторінках за допомогою HTML або JavaScript.

2.1.3 HTML, CSS, та JavaScript

- HTML необхідний для будь-якого проекту веб-розробки, оскільки він є основою відтворення веб-вмісту.
- CSS надає можливість створювати візуально привабливий і послідовний дизайн веб-сайту.
- JavaScript потрібен для додавання динамічних елементів і взаємодії в реальному часі у веб-додатку, наприклад оновлення вмісту без перезавантаження сторінки.

Документи HTML структуровані як ряд елементів, кожен з яких визначається тегами. Ці елементи визначають вміст і структуру веб-сторінок,

утворюючи скелет, який потім стилізується за допомогою CSS і маніпулюється JavaScript.

CSS дозволяє веб-розробникам застосовувати стилі до веб-елементів за допомогою селекторів і властивостей. Ці стилі можуть включати шрифти, кольори, макети та переходи, серед іншого, що робить інтерфейс користувача більш привабливим і функціональним.

CSS безпосередньо змінює представлення HTML-вмісту, і його можна посилати зовні або включати в документи HTML. Зміни в інтерфейсі користувача, необхідні логіці на стороні сервера (через Flask) або сценарію на стороні клієнта (через JavaScript), можуть бути застосовані за допомогою динамічних маніпуляцій CSS.

JavaScript працює в браузері клієнта та забезпечує динамічні зміни вмісту HTML та CSS. Він може надсилати асинхронні запити на Flask сервер, обробляти вибірку, публікацію або оновлення даних і відображати зміни на веб-сторінці без перезавантаження.

JavaScript покращує взаємодію користувача з HTML-формами, кнопками та іншими елементами інтерфейсу, обробляючи події та запускаючи дії. Він отримує дані з SQLite через Flask, динамічно оновлюючи веб-вміст на основі взаємодії користувача або зміни даних.

2.2 Вибір технологій розробки чат-бота-консультанта

Для розробки чат-бота-консультанта я обрав AssistantAPI від OpenAI. Це рішення обумовлене його здатністю базуватись на контексті та заданих інструкціях при відносній легкості налаштування, що дозволяє швидко адаптувати модель під специфіку мого веб-додатку з високою точністю та релевантністю відповідей.

AssistantAPI інтегрується до Flask через API-запити, що дозволяє надсилати повідомлення на сервер OpenAI та отримувати відповіді, і налаштовується через веб-інтерфейс Playground.

Давайте детально розглянемо інтегрування AssistantAPI у Flask та його подальше налаштування у веб-інтерфейсі Playground:

Інтегрування AssistantAPI у Flask:

1. Спочатку потрібно встановити бібліотеку OpenAI:

```
pip install flask openai
```

2. Далі ваш основний файл Flask-додатку повинен мати API ключ. Його можна зберігати у файлі .env для безпеки. API-ключі генеруються на сайті OpenAI

```
import os
from dotenv import load_dotenv
load_dotenv()
OPENAI_API_KEY = os.getenv('OPENAI_API_KEY')
openai.api_key = 'your_openai_api_key'
```

3. Тепер потрібно створити функцію, яка буде викликати Assistant API:

```
import openai
openai.api_key = OPENAI_API_KEY
def get_assistant_response(prompt):
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=prompt,
        max_tokens=150 )
    return response.choices[0].text.strip()
```

4. Створимо маршрут, який буде отримувати запити від користувачів та використовувати Assistant API для генерації відповідей:

```
@app.route('/chat', methods=['POST'])
def chat():
    data = request.get_json()
    prompt = data.get('prompt', '')
    if prompt:
        response_text = get_assistant_response(prompt)
        return jsonify({'response': response_text})
    else:
        return jsonify({'error': 'No prompt provided'}), 400
```

Налаштування AssistantAPI здійснюється через веб-інтерфейс Playground, який дозволяє тестувати та налаштовувати модель у реальному часі. У Playground можна задавати контекст, формулювати інструкції та експериментувати з різними параметрами для оптимізації роботи чат-бота-консультанта (рис. 2.1).

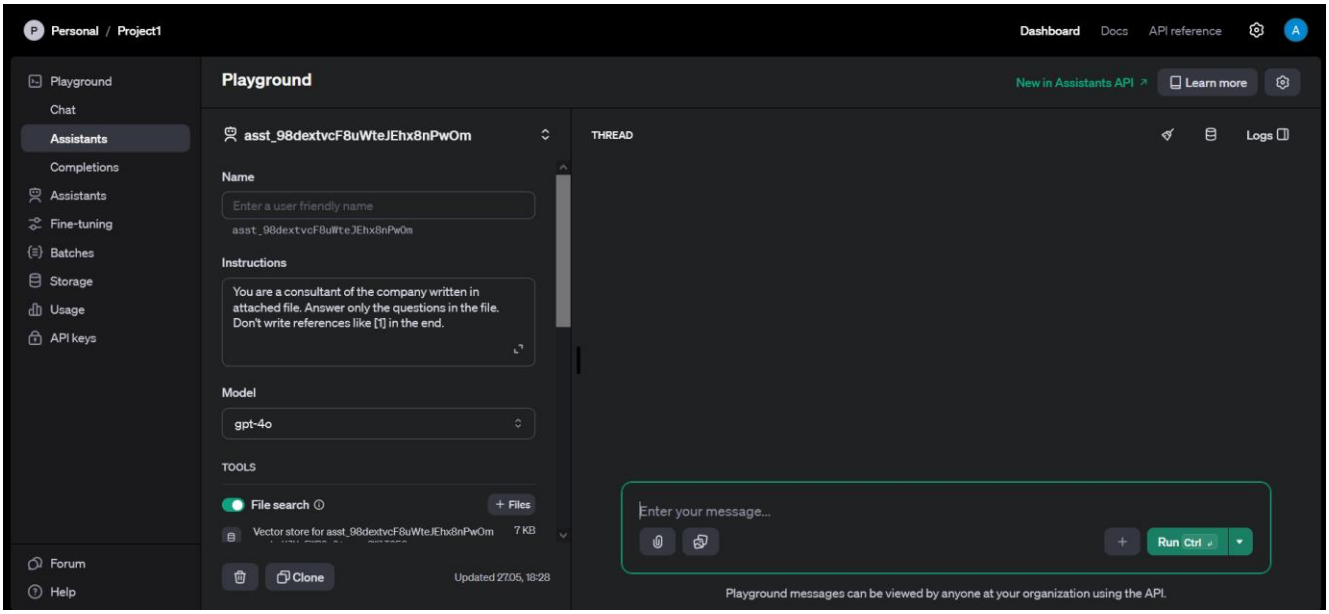


Рис. 2.1 Веб-інтерфейс Playground

Для мого проєкту особливо цікава здатність моделі аналізувати та навчатись на інформації, довільно завантаженої до неї, оскільки це дозволятиме генерувати відповіді на запитання користувачів, відштовхуючись від наданого контексту. Ця функція називається “File search”, яка дозволяє додавати контексну інформацію для її подальшого аналізу моделлю, на яку AssistantAPI буде постійно посилатися, відповідаючи на питання користувачів (рис. 2.2). У нашому випадку, оскільки я створюю бібліотеку з купою різноманітних книг, я завантажив до AssistantAPI базу даних, яка налічує всі наявні книги та їх інформацію, доступну на веб-сайті (рис 2.3, 2.4). Окрім цього, я завантажив до AssistantAPI загальну інформацію о бібліотеці та контактну інформацію для зворотного зв’язку з компанією.

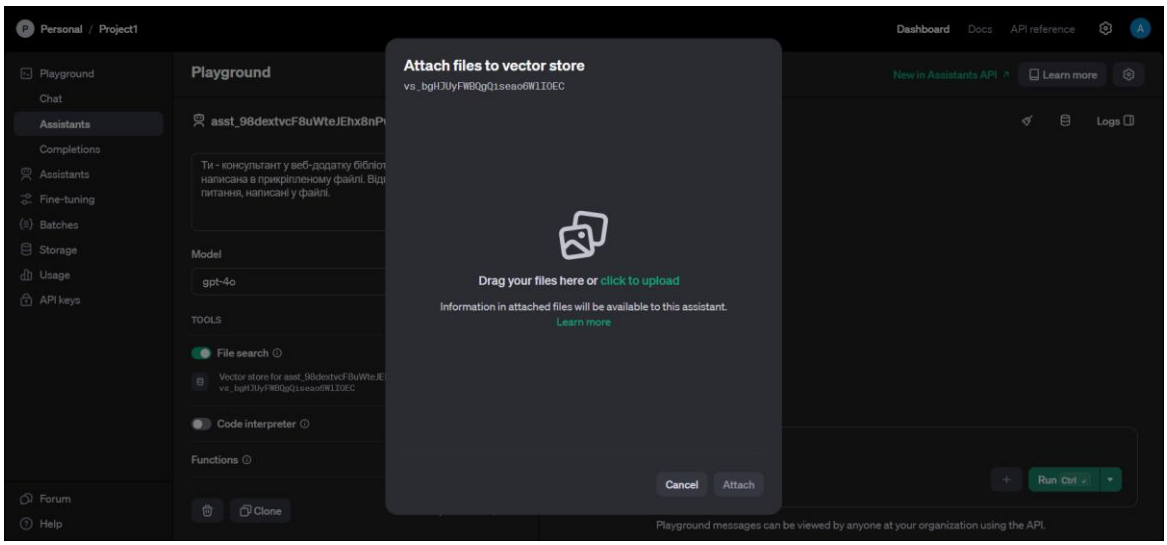


Рис. 2.2 Завантаження файлу в File Search

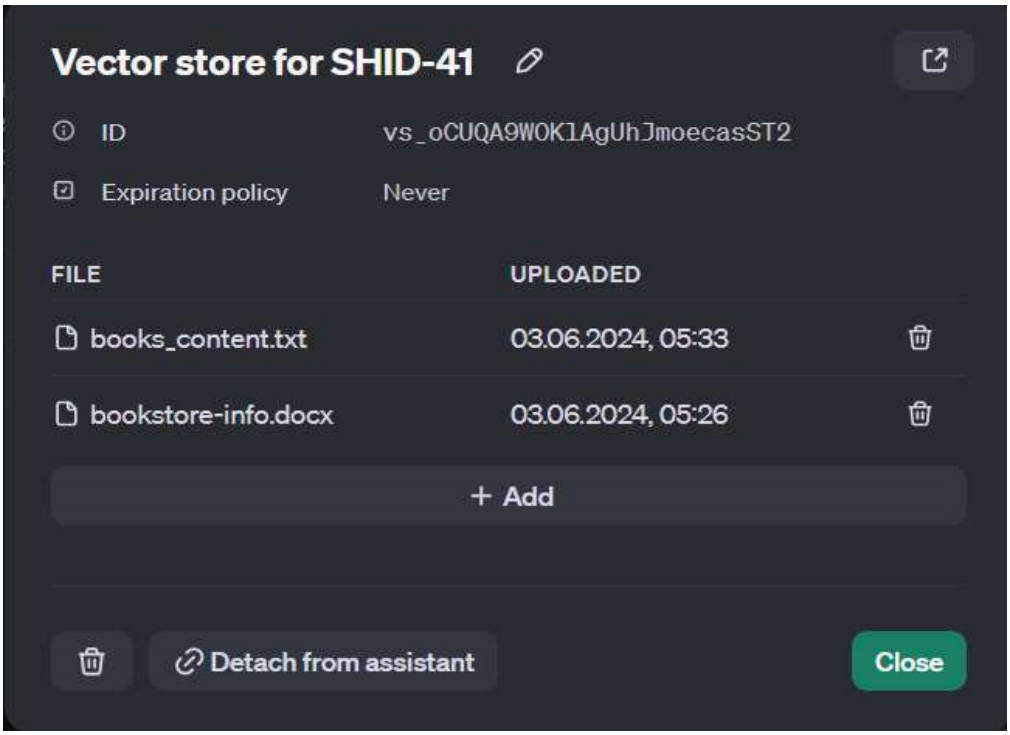


Рис 2.3 Завантаження файлів з інформацією

books_content.txt - Notepad2

File Edit View Settings ?

```
book_id title author  price publication_year  publisher_id
112107  Чиста архітектура   Роберт Сесіл Мартін   60.0  2022  3141375
123554  PYTHON для дітей. Веселий вступ до програмування  Бріґґс Дж 390.0 2020  3141375
146791  Над прірвою у житі  Джером Д.Сэлінджер  130.0 1951  2955415
150402  Think Java: How to Think Like a Computer Scientist  Аллен Дауні, Кріс Мейфілд 130.0 2024  3141375
162350  Курочка Ряба  Володимир Харченко  94.0  2003  3141375
174448  Learn Python the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code John M. Zelle 299.0 2019  3141375
198025  Thinking in Java  Bruce Eckel 320.0 2019  3141375
251788  Кіндзадза Данелія Георгій Миколайови  198.0 1896  3141375
262154  Java. Бібліотека професіонала Кей С. Хорстманн, Гарі Корнелл  140.0 2020  3141375
272956  Книга Пришвидшений курс Python. Практичний, проєктно-орієнтований вступ до програмування  Ерік Маттес   500.0 2023  3141375
307069  Книга Head First Java. Легкий для сприйняття довідник Кетті Сьєрра, Берт Бейтс  173.0 2020  3141375
327296  Java For Dummies  Barry A. Burd 230.0 2016  3141375
438461  Дванадцять стільців Євген Петров, Ілля Ільф 100.0 1993  3141375
679961  Грокаємо алгоритми. Ілюстрований посібник для програмістів і допитливих Адітья Бхаргава 420.0 2021  3141375
838880  Java. Керівництво для початківців Герберт Шілдт 78.0  2023  3141375
907058  Основи программування для абсолютних початківців  Натан Кларк 223.0 2017  3141375
```

Рис. 2.4 База даних книг

Тепер, коли ми додали до AssistantAPI контексту інформацію як орієнтир, нам потрібно задати інструкції для того, щоб модель розуміла, що з цією інформацією потрібно робити та як себе потрібно поводити в тих чи інших випадках. Інструкції задаються у вікні Instructions (рис 2.3). У нашому випадку, асистент отримує наступну інструкцію: "Ви ввічливий і досвідчений консультант бібліотеки 'ШІД-41' з питань довідкової інформації бібліотеки. Використовуйте надані документи як базу знань, щоб відповісти на запитання. Відповідайте лише на питання, надані в документі."

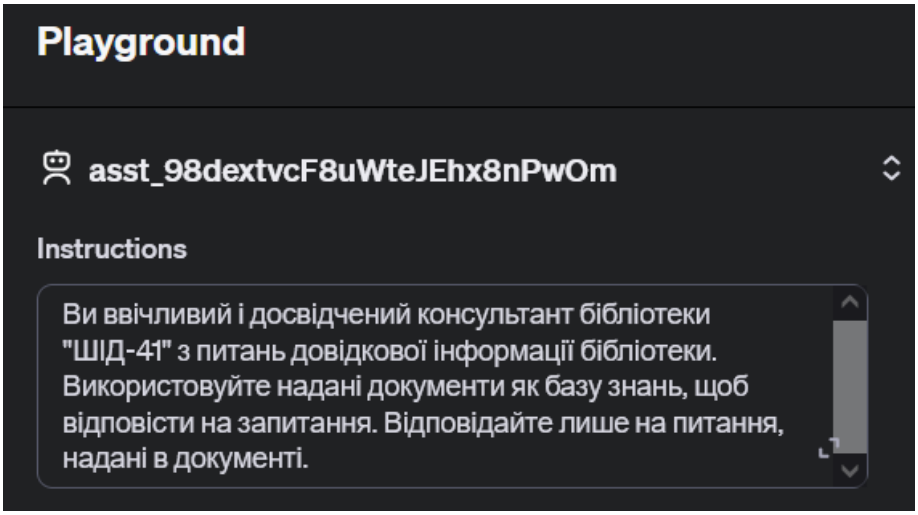


Рис. 2.3 Задання інструкцій

Таким чином, використання AssistantAPI значно підвищує ефективність комунікацій між користувачами та компанією, завдяки можливості моментально відповідати на загальні питання о додатку.

2.3 Проектування системи

Проектування веб-додатку є ключовим етапом у створенні програмного забезпечення, оскільки воно визначає архітектуру, основні компоненти та взаємодію між ними.

2.3.1 Основні компоненти веб-додатку

Розроблений веб-додаток містить:

1) Main.py:

– Основний файл, який запускає Flask сервер.

- Містить маршрути для обробки різних HTTP-запитів (GET, POST);
- Взаємодіє з Assistant API (assistant_api.py) через функцію create_assistant(client);
- Використовує шаблони для відображення HTML-сторінок;
- Взаємодіє з JavaScript (script.js) для функціонування чат-бота на стороні клієнта;
- Підключається до бази даних SQLite (bookstore.db);
- Виконує CRUD операції над таблицями бази даних (додавання нових записів, оновлення існуючих, видалення та відображення даних з таблиць Books, Customers, Orders, Publishers. Books).

2) Assistant_api.py:

- Відповідає за створення та налаштування асистента через API OpenAI.

3) Script.js:

- Взаємодіє з Flask через HTTP-запити до маршрутів /start та /chat.
- Відповідає за інтерактивність чат-бота, обробку подій та відправку/отримання повідомлень через API.

4) Style.css:

- Стилізує HTML сторінки та елементи інтерфейсу, включаючи чат-бота.

5) HTML (index.html, orders.html, books.html, customers.html):

- Шаблони HTML, які відображають дані, отримані з Flask серверу.
- Взаємодіють з CSS для оформлення і JavaScript для функціональності.

6) Bookstore.db:

- Зберігає інформацію про книги, клієнтів, замовлення та видавців;
- Використовується для виконання CRUD операцій через Flask маршрути (додавання нових записів, оновлення існуючих, видалення та відображення даних з таблиць Books, Customers, Orders, Publishers. Books).

2.3.2 Діаграма класів

Об’єктно-орієнтовна модель компонентів додатку була розроблена за допомогою веб-сайту Plantuml (рис. 2.4).

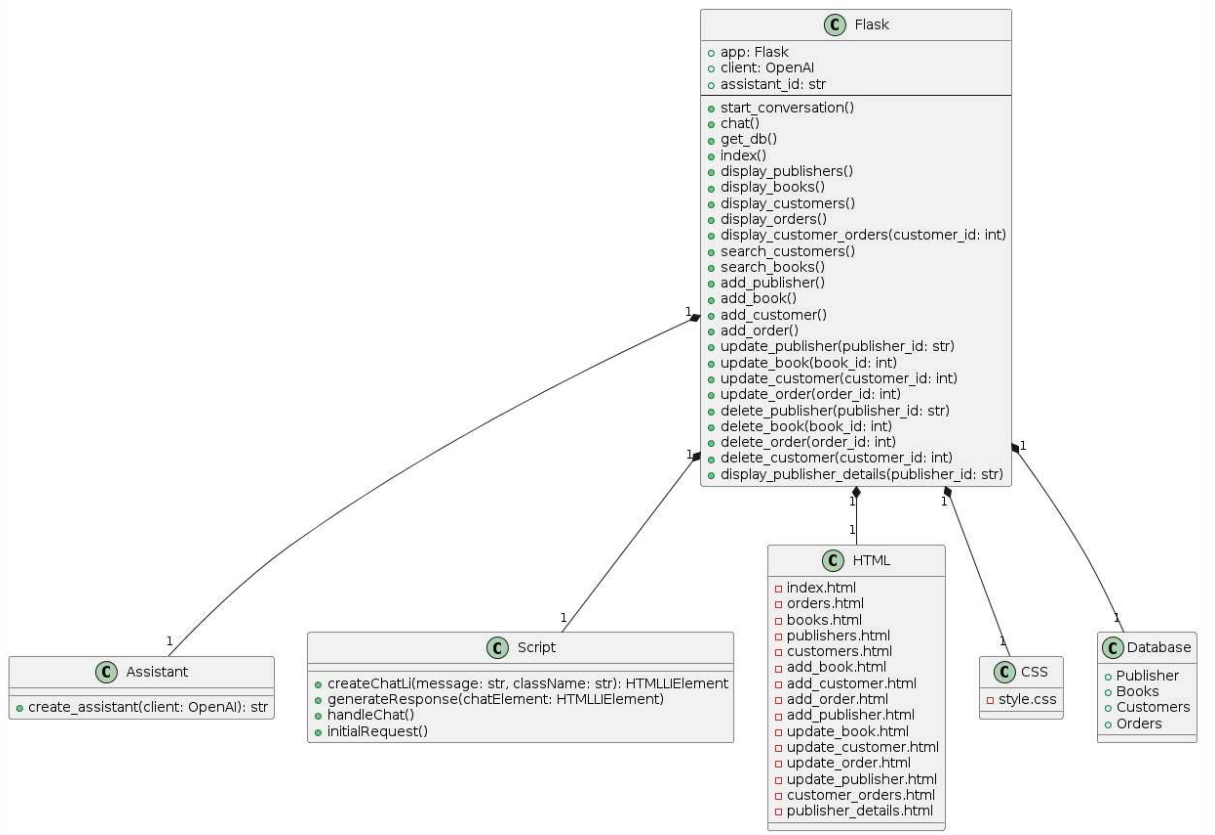


Рис 2.4 Діаграма класів

2.3.3 Діаграма послідовностей

Діаграма послідовностей (рис. 2.5) представляє собою процес обробки HTTP-запиту через сервер Flask.

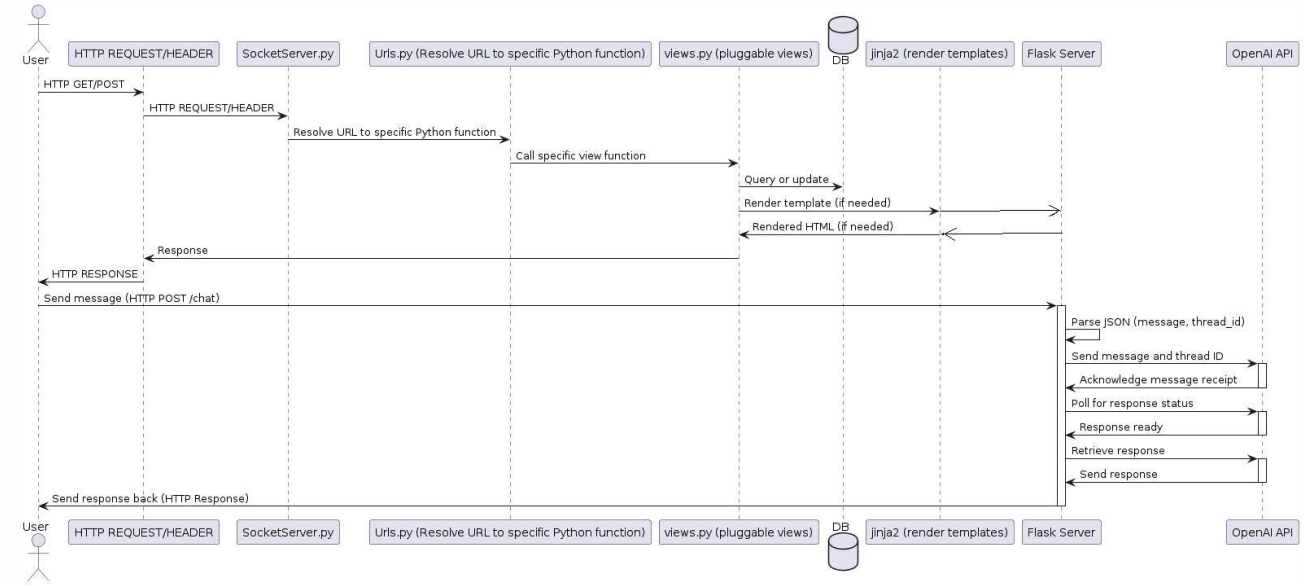


Рис. 2.5 Діаграма послідовностей

Давайте детально розглянемо процес обробки HTTP-запитів у Flask та обробку повідомлень OpenAI API:

1) Користувач через браузер ініціює запит шляхом доступу до веб-сторінки (HTTP GET), або шляхом надсилання форми (HTTP POST);

2) Обробка HTTP-запиту:

HTTP REQUEST/HEADER: представляє початкову точку входу, де отримано HTTP-запит (GET чи POST);

3) SocketServer.py:

SocketServer.py є частиною стандартної бібліотеки Python для створення серверів, які обробляють низькорівневі мережеві з'єднання, такі як прийом і відправка даних по мережі. Він отримує HTTP запити від клієнта і передає їх на обробку вищого рівня URL маршрутизатора Flask. Коли клієнт відправляє HTTP запит до сервера, SocketServer.py приймає цей запит і аналізує його заголовки.

4) Urls.py перетворює URL-адресу вхідного запиту на конкретну функцію перегляду, визначену в програмі. Саме тут механізм маршрутизації Flask зіставляє URL із відповідною функцією перегляду.

5) Views.py містить визначення функцій обробки запитів у Flask. Коли URL запит співпадає з маршрутом, визначеним у views.py, викликається відповідна функція обробки. Ця функція запитує або оновлює дані з бази даних, після чого надає цю інформацію для рендерингу Jinja2.

6) Database представляє базу даних SQLite, у якій зберігаються дані програми. Функції views.py взаємодіють з базою даних для запиту або оновлення даних.

7) Jinja2 є механізмом шаблонів для Python, який використовується у Flask для динамічного рендерингу шаблонів HTML. Дані, отримані від views.py, передаються шаблонам Jinja2 для генерації остаточної відповіді HTML.

- Функція обробки у views.py викликає render_template, передаючи ім'я шаблону та дані, які потрібно включити у шаблон;
- Jinja2 обробляє шаблон, замінюючи змінні їх значеннями та виконуючи логіку шаблону;

– Згенерований HTML передається до Flask серверу.

8) Після цього згенерований HTML-контент через сервер Flask відправляється назад користувачеві.

9) Чат з OpenAI API:

Взаємодія з користувачем:

– Користувач надсилає запит HTTP POST до кінцевої точки /chat із повідомленням і ідентифікатором потоку;
– Flask Server отримує запит і аналізує корисне навантаження JSON, щоб отримати повідомлення та ідентифікатор потоку;
– Flask Server надсилає повідомлення користувача OpenAI API за допомогою екземпляра клієнта;
– OpenAI API отримує повідомлення та обробляє його;
– OpenAI API надсилає підтвердження отримання повідомлення на сервер Flask;
– Flask Server опитує OpenAI API статус генерації відповіді;
– OpenAI API обробляє повідомлення та генерує відповідь;
– API OpenAI інформує сервер Flask, що відповідь готова;
– Flask Server отримує відповідь від OpenAI API;
– Flask Server повертає користувачеві HTTP відповідь.

2.3.4 Діаграма прецедентів

Згідно з ISO 24765, проектування визначається як процес, який включає визначення архітектури, компонентів, інтерфейсів та інших необхідних характеристик системи. Результатом цього процесу є детально розроблений проект, що містить повний набір моделей, властивостей і характеристик, представлених у форматі, що дозволяє реалізувати систему.

Діаграма варіантів використання в UML, також відома як діаграма прецедентів, візуально показує зв'язки між акторами та варіантами використання, будучи ключовою частиною моделі прецедентів. Вона дозволяє описувати систему на концептуальному рівні. Основні елементи такої діаграми включають:

– актора, який є зовнішньою сутністю щодо системи, здатною взаємодіяти з нею, зазвичай зображується як стилізована фігурка людини;
– прецедент (варіант використання), що представляє дії системи, які призводять до результатів, помітних для акторів, і відображається у формі еліпса з відповідним написом.

Для налагодження взаємозв'язків між акторами та прецедентами в UML використовуються декілька стандартних типів зв'язків. Асоціативні відносини (association relationship) допомагають позначити конкретну роль актора у взаємодії з певним прецедентом.

Для створення діаграми варіантів використання необхідно виконати наступні кроки:

1. Визначити акторів – групи осіб, що взаємодіють з системою, використовуючи різні рівні доступу.

2. Ідентифікувати якомога більше варіантів використання, що охоплюють можливі процеси, які можуть виконувати актори.

Для розроблювального веб-додатку акторами є адміністратор і користувач. Діаграму прецедентів наведено на рис. 2.6.

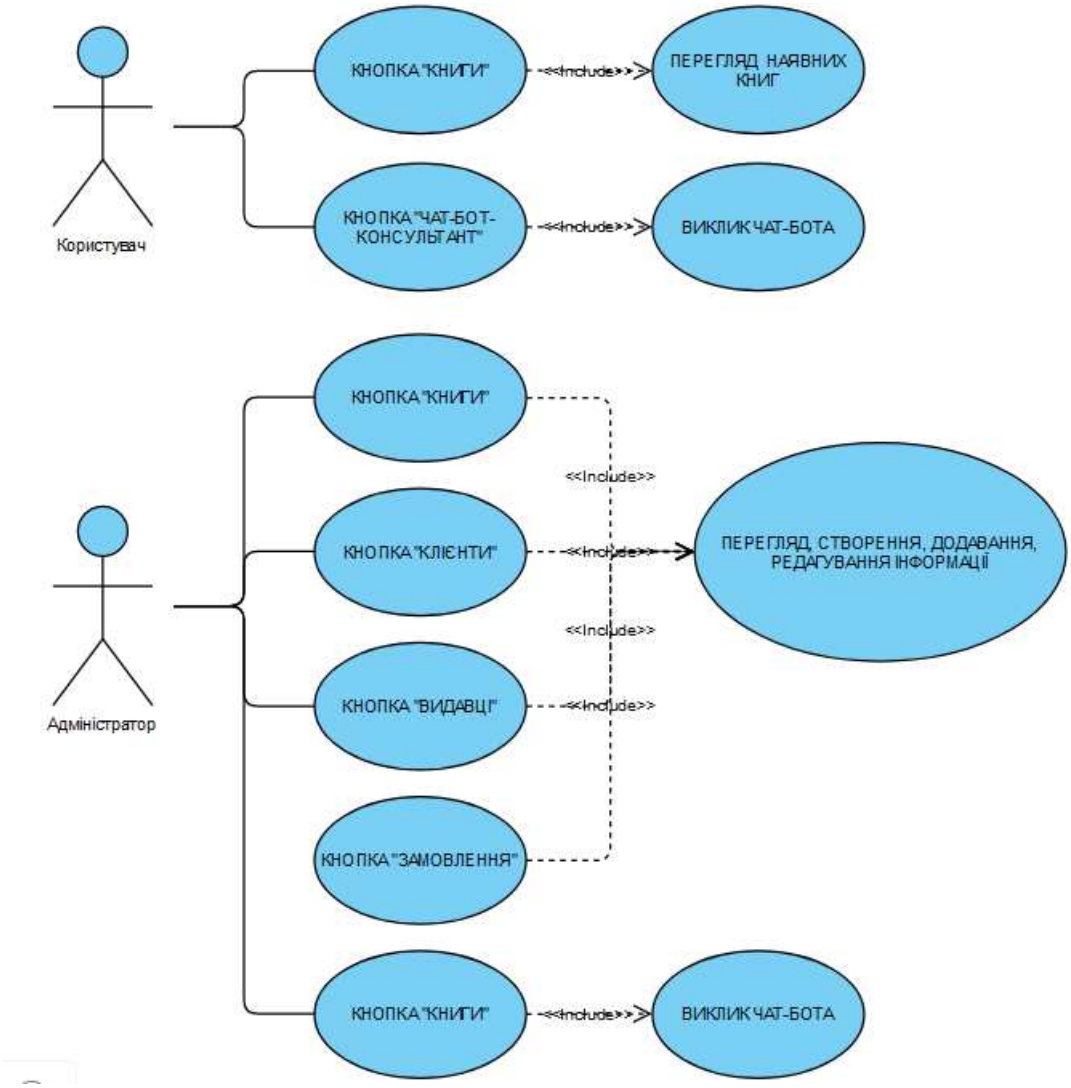


Рис. 2.6 Діаграма прецедентів веб-додатку

Ставлення включення (include relationship) між двома варіантами використання - вказує на те, що задана поведінка для одного варіанту використання включається як складовий фрагмента в послідовність поведінки іншого варіанту використання.

Ставлення розширення (extend relationship) - визначає взаємозв'язок базового варіанту використання з іншим варіантом використання, функціональне поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Ставлення узагальнення (generalization relationship) між варіантами використання застосовується в тому випадку, коли необхідно відзначити, що

дочірні варіанти використання мають усі особливості поведінки батьківських варіантів.

Актор «Адміністратор» – має повний доступ до системи і відповідає за її функціонування в цілому. Варіанти використання для актора «Адміністратор» наведені у таблиці 2.1.

Таблиця 2.1

Варіанти використання для актора «Адміністратор»

Найменування	Опис
Кнопка "Книги"	Натискаючи на цю кнопку, адміністратор може переглядати, додавати, редагувати та видаляти інформацію про книги.
Кнопка "Клієнти"	Натискаючи на цю кнопку, адміністратор може переглядати, додавати, редагувати та видаляти інформацію про клієнтів.
Кнопка "Видавці"	Натискаючи на цю кнопку, адміністратор може переглядати, додавати, редагувати та видаляти інформацію про видавців.
Кнопка "Замовлення"	Натискаючи на цю кнопку, адміністратор може переглядати, додавати, редагувати та видаляти замовлення.
Кнопка "Чат-бот-консультант"	Натискаючи на цю кнопку, адміністратор може взаємодіяти з чат-ботом для отримання допомоги або консультацій.

Актор «Користувач» – клієнт бібліотеки, який може переглядати книги та використовувати чат-бота-консультанта. Варіанти використання для актора «Користувач» наведені у таблиці 2.2.

Таблиця 2.2

Варіанти використання для актора «Користувач»

Найменування	Опис
Кнопка "Книги"	Натискаючи на цю кнопку, користувач може переглядати всі наявні книги бібліотеки.
Кнопка "Чат-бот-консультант"	Натискаючи на цю кнопку, користувач може взаємодіяти з чат-ботом для отримання консультацій або допомоги.

Опис сценаріїв використання:

Сценарій 1: Перегляд книг користувачем

Актор: Користувач

Передумови: Користувач знаходиться на головній сторінці веб-додатку.

Основний потік:

– Користувач натискає на кнопку "Книги";
– Система відображає список всіх наявних книг бібліотеки;
– Користувач переглядає книги.

Сценарій 2: Виклик чат-бота користувачем

Актор: Користувач

Передумови: Користувач знаходиться на головній сторінці веб-додатку.

Основний потік:

– Користувач натискає на кнопку "Чат-бот-консультант";
– Система відкриває вікно чат-бота;
– Користувач вводить питання або запит;
– Чат-бот відповідає на питання або надає допомогу.

Сценарій 3: Управління книгами адміністратором

Актор: Адміністратор

Передумови: Адміністратор знаходиться на головній сторінці веб-додатку.

Основний потік:

- Адміністратор натискає на кнопку "Книги";
- Система відображає список всіх наявних книг бібліотеки;
- Адміністратор може додавати нові книги, редагувати інформацію про існуючі книги або видаляти книги.

Сценарій 4: Управління клієнтами адміністратором

Актор: Адміністратор

Передумови: Адміністратор знаходиться на головній сторінці веб-додатку.

Основний потік:

- Адміністратор натискає на кнопку "Клієнти";
- Система відображає список всіх клієнтів бібліотеки;
- Адміністратор може додавати нових клієнтів, редагувати інформацію про існуючих клієнтів або видаляти клієнтів.

Сценарій 5: Управління видавцями адміністратором

Актор: Адміністратор

Передумови: Адміністратор знаходиться на головній сторінці веб-додатку.

Основний потік:

- Адміністратор натискає на кнопку "Видавці";
- Система відображає список всіх видавців;
- Адміністратор може додавати нових видавців, редагувати інформацію про існуючих видавців або видаляти видавців.

Сценарій 6: Управління замовленнями адміністратором

Актор: Адміністратор

Передумови: Адміністратор знаходиться на головній сторінці веб-додатку.

Основний потік:

- Адміністратор натискає на кнопку "Замовлення";
- Система відображає список всіх замовлень;
- Адміністратор може додавати нові замовлення, редагувати інформацію про існуючі замовлення або видаляти замовлення.

Сценарій 7: Виклик чат-бота адміністратором

Актор: Адміністратор

Передумови: Адміністратор знаходиться на головній сторінці веб-додатку.

Основний потік:

– Адміністратор натискає на кнопку "Чат-бот-консультант";
– Система відкриває вікно чат-бота;
– Адміністратор вводить питання або запит;
– Чат-бот відповідає на питання або надає допомогу.

3 РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ТА ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Пояснення основного коду

В цьому розділі я детально розберу функціональність коду основного файлу, з якого запускається Flask сервер:

Завантаження .env, в якому міститься api-ключ

load_dotenv()

Зчитування api ключа OpenAI

api_key = os.getenv("OPENAI_API_KEY")

Ініціалізується програма Flask

app = Flask(__name__, '/static/')

Увімкнення спільного використання ресурсів між джерелами (CORS) для Flask для підтримки облікових даних

CORS(app, supports_credentials=True)

Ініціація OpenAI клієнта

client = OpenAI()

Створення нового помічника або завантаження наявного

assistant_id = create_assistant(client)

Підключення до бази даних

def get_db():

db = getattr(g, '_database', None)

if db is None:

db = g._database = SQLite.connect('bookstore.db')

return db

Створення таблиць в базі даних (якщо вже не існують)

with app.app_context():

conn = get_db()

cursor = conn.cursor()

cursor.execute('"CREATE TABLE IF NOT EXISTS Publisher (

publisher_id TEXT PRIMARY KEY,

```
publisher_name TEXT,
publication_address TEXT,
publication_phone TEXT,
publication_web TEXT)''')
cursor.execute('''CREATE TABLE IF NOT EXISTS Books (
book_id INTEGER PRIMARY KEY AUTOINCREMENT,
title TEXT,
author TEXT,
price REAL,
publication_year INTEGER,
FOREIGN KEY (publisher_id) REFERENCES Publisher (publisher_id) )''')
cursor.execute('''CREATE TABLE IF NOT EXISTS Publisher_Books_Junction (
publisher_id TEXT,
book_id INTEGER,
FOREIGN KEY (publisher_id) REFERENCES Publisher (publisher_id),
FOREIGN KEY (book_id) REFERENCES Books (book_id),
PRIMARY KEY (publisher_id, book_id))''')
cursor.execute('''CREATE TABLE IF NOT EXISTS Customers (
customer_id INTEGER PRIMARY KEY,
name TEXT, email TEXT, phone TEXT)''')
cursor.execute('''CREATE TABLE IF NOT EXISTS Orders (
order_id INTEGER PRIMARY KEY,
customer_id INTEGER,
book_id INTEGER,
quantity INTEGER,
order_date TEXT,
Total_price INTEGER,
FOREIGN KEY (customer_id) REFERENCES Customers (customer_id),
FOREIGN KEY (book_id) REFERENCES Books (book_id))''')
conn.commit()
```

```
cursor.close()
conn.close()
```

Домашня сторінка

```
@app.route('/')
def index(): return render_template('index.html')
```

Відображення всіх видавців

```
@app.route('/publishers')
def display_publishers():
conn = get_db()
cursor = conn.cursor()
cursor.execute('SELECT * FROM Publisher')
publishers = cursor.fetchall()
return render_template('publishers.html', publishers=publishers)
```

Відображення всіх книг

```
@app.route('/books')
def display_books():
conn = get_db()
cursor = conn.cursor()
cursor.execute('SELECT * FROM Books')
books = cursor.fetchall()
return render_template('books.html', books=books)
```

Відображення всіх клієнтів

```
@app.route('/customers')
def display_customers():
conn = get_db()
cursor = conn.cursor()
cursor.execute('SELECT * FROM Customers')
customers = cursor.fetchall()
return render_template('customers.html', customers=customers)
```

Відображення всіх замовлень

```
@app.route('/orders')
def display_orders():
conn = get_db()
cursor = conn.cursor()
cursor.execute('''SELECT Orders.order_id, Customers.customer_id,
Customers.name, Books.title, Orders.quantity, Orders.order_date, Orders.Total_Price
FROM Orders
JOIN Customers ON Orders.customer_id = Customers.customer_id
JOIN Books ON Orders.book_id = Books.book_id''')
orders = cursor.fetchall()
return render_template('orders.html', orders=orders)
```

Відображення всіх замовлень по клієнтах

```
@app.route('/customers/orders/<int:customer_id>')
def display_customer_orders(customer_id):
conn = get_db()
cursor = conn.cursor()
cursor.execute('''SELECT Orders.order_id, Customers.customer_id,
Customers.name, Books.title, Orders.quantity, Orders.order_date, Orders.Total_price
FROM Orders
JOIN Customers ON Orders.customer_id = Customers.customer_id
JOIN Books ON Orders.book_id = Books.book_id
WHERE Customers.customer_id = ?''', (customer_id,))
customer_orders = cursor.fetchall()
return render_template('customer_orders.html',
customer_orders=customer_orders)
```

Пошук клієнтів по імені та email

```
@app.route('/customers/search', methods=['GET'])
def search_customers():
conn = get_db()
cursor = conn.cursor()
```

search_query = request.args.get('query', '').strip()

if search_query:

Виконання пошукового запиту

cursor.execute('SELECT * FROM Customers WHERE name LIKE ? OR email LIKE ?',

('%' + search_query + '%', '%' + search_query + '%'))

customers = cursor.fetchall()

else:

Якщо пошуковий запит не вказано, поверніть усіх клієнтів

cursor.execute('SELECT * FROM Customers')

customers = cursor.fetchall()

return render_template('customers.html', customers=customers)

Шукати книги

@app.route('/books/search', methods=['GET'])

def search_books():

search_query = request.args.get('query', '').strip()

if not search_query:

Якщо пошуковий запит порожній, відобразити всі книги

return redirect('/books')

conn = get_db()

cursor = conn.cursor()

Виконання пошукового запиту за допомогою SQL LIKE для фільтрації результатів

cursor.execute('SELECT * FROM Books WHERE book_id LIKE ? OR title LIKE ? OR author LIKE ?', ('%' + search_query + '%', '%' + search_query + '%', '%' + search_query + '%'))

books = cursor.fetchall()

return render_template('books.html', books=books)

Добавити нового видавця

@app.route('/publishers/add', methods=['GET', 'POST'])

```
def add_publisher():
if request.method == 'POST':
conn = get_db()
cursor = conn.cursor()
publisher_name = request.form['publisher_name']
publication_address = request.form['publication_address']
publication_phone = request.form['publication_phone']
publication_web = request.form['publication_email']
```

Створення 6-значного унікального ідентифікатора видавця

```
publisher_id = str(uuid.uuid4().int)[:7]
cursor.execute('INSERT INTO Publisher (publisher_id, publisher_name,
publication_address, publication_phone, publication_web) VALUES (?, ?, ?, ?, ?)',
(publisher_id, publisher_name, publication_address, publication_phone,
publication_web))
conn.commit()
return redirect('/publishers')
return render_template('add_publisher.html')
```

Добавити книгу

```
@app.route('/books/add', methods=['GET', 'POST'])
def add_book():
if request.method == 'POST':
conn = get_db()
cursor = conn.cursor()
title = request.form['title']
author = request.form['author']
price = request.form['price']
publication_year = request.form['publication_year']
publisher_id = request.form['publisher_id']
```

Створення 6-значного унікального ідентифікатора видавця

```
book_id = str(uuid.uuid4().int)[:6]
```

```
        cursor.execute('INSERT INTO Books (book_id, title, author, price,
publication_year, publisher_id) VALUES (?, ?, ?, ?, ?, ?)',
        (book_id, title, author, price, publication_year, publisher_id))
        conn.commit()
        return redirect('/books')
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT publisher_id, publisher_name FROM Publisher')
        publishers = cursor.fetchall()  # Fetch all the publishers to display in the
dropdown select field
        return render_template('add_book.html', publishers=publishers)
```

Добавити нового кліента

```
        @app.route('/customers/add', methods=['GET', 'POST'])
        def add_customer():
        if request.method == 'POST':
        conn = get_db()
        cursor = conn.cursor()
        name = request.form['name']
        email = request.form['email']
        phone = request.form['phone']
```

Створення 10-значного унікального ідентифікатора видавця

```
        customer_id = str(uuid.uuid4().int)[:10]
        cursor.execute('INSERT INTO Customers (customer_id, name, email, phone)
VALUES (?, ?, ?, ?)', (customer_id, name, email, phone))
        conn.commit()
        return redirect('/customers')
        return render_template('add_customer.html')
```

Добавити нове замовлення

```
        @app.route('/orders/add', methods=['GET', 'POST'])
        def add_order():
```

```
if request.method == 'POST':
conn = get_db()
cursor = conn.cursor()
customer_id = request.form['customer_id']
book_id = request.form['book_id']
quantity = request.form['quantity']
order_date = request.form['order_date']
```

Створення унікального ідентифікатора для замовлення

```
order_id = str(uuid.uuid4().int)[:8]
```

Отримання ціни книги

```
cursor.execute('SELECT price FROM Books WHERE book_id = ?', (book_id,))
book_price = cursor.fetchone()[0]
```

Розрахунок ціни замовлення

```
order_price = float(quantity) * float(book_price)
cursor.execute('INSERT INTO Orders (order_id, customer_id, book_id, quantity,
order_date, Total_price) VALUES (?, ?, ?, ?, ?, ?)',
(order_id, customer_id, book_id, quantity, order_date, order_price))
conn.commit()
return redirect('/orders')
conn = get_db()
cursor = conn.cursor()
cursor.execute('SELECT * FROM Customers')
customers = cursor.fetchall()
cursor.execute('SELECT * FROM Books')
books = cursor.fetchall()
return render_template('add_order.html', customers=customers, books=books)
```

Оновити видавця

```
@app.route('/publishers/update/<string:publisher_id>', methods=['GET', 'POST'])
def update_publisher(publisher_id):
conn = get_db()
```

```
cursor = conn.cursor()
if request.method == 'POST':
publisher_name = request.form['publisher_name']
publication_address = request.form['publication_address']
publication_phone = request.form['publication_phone']
publication_web = request.form['publication_web']
cursor.execute('UPDATE Publisher SET publisher_name=?,
publication_address=?, publication_phone=?, publication_web=? WHERE
publisher_id=?', (publisher_name, publication_address, publication_phone,
publication_web, publisher_id))
conn.commit()
return redirect('/publishers')
cursor.execute('SELECT * FROM Publisher WHERE publisher_id=?',
(publisher_id,))
publisher = cursor.fetchone()
return render_template('update_publisher.html', publisher=publisher,
publisher_id=publisher_id)
```

Оновити книгу

```
@app.route('/books/update/<int:book_id>', methods=['GET', 'POST'])
def update_book(book_id):
conn = get_db()
cursor = conn.cursor()
if request.method == 'POST':
title = request.form['title']
author = request.form['author']
price = request.form['price']
publication_year = request.form['publication_year']
publisher_id = request.form['publisher_id']  # Get the selected publisher_id from
the form
cursor.execute('UPDATE Books SET title=?, author=?, price=?,
```

```
publication_year=?, publisher_id=? WHERE book_id=?',
        (title, author, price, publication_year, publisher_id, book_id))
        conn.commit()
        return redirect('/books')
        cursor.execute('SELECT * FROM Books WHERE book_id=?', (book_id,))
        book = cursor.fetchone()
        cursor.execute('SELECT publisher_id, publisher_name FROM Publisher')
        publishers = cursor.fetchall()  # Fetch all the publishers to display in the
dropdown select field
        return render_template('update_book.html', book=book, publishers=publishers)
```

Оновити кліента

```
        @app.route('/customers/update/<int:customer_id>', methods=['GET', 'POST'])
        def update_customer(customer_id):
        conn = get_db()
        cursor = conn.cursor()
        if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        phone = request.form['phone']
        cursor.execute('UPDATE Customers SET name=?, email=?, phone=? WHERE
customer_id=?', (name, email, phone, customer_id))
        conn.commit()
        return redirect('/customers')
        cursor.execute('SELECT * FROM Customers WHERE customer_id=?',
(customer_id,))
        customer = cursor.fetchone()
        return render_template('update_customer.html', customer=customer,
customer_id=customer_id)
```

Оновити замовлення

```
        @app.route('/orders/update/<int:order_id>', methods=['GET', 'POST'])
```

```
def update_order(order_id):
conn = get_db()
cursor = conn.cursor()
if request.method == 'POST':
customer_id = request.form['customer_id']
book_id = request.form['book_id']
quantity = request.form['quantity']
order_date = request.form['order_date']
```

Отримати ціну книги

```
cursor.execute('SELECT price FROM Books WHERE book_id = ?', (book_id,))
book_price = cursor.fetchone()[0]
```

Розрахувати ціну замовлення

```
order_price = float(quantity) * float(book_price)
cursor.execute('UPDATE Orders SET customer_id=?, book_id=?, quantity=?,
order_date=?, Total_price=? WHERE order_id=?',
(customer_id, book_id, quantity, order_date, order_price, order_id))
conn.commit()
return redirect('/orders')
cursor.execute('''SELECT Orders.order_id, Customers.name, Books.title,
Orders.quantity, Orders.order_date, Orders.Total_price
FROM Orders
JOIN Customers ON Orders.customer_id = Customers.customer_id
JOIN Books ON Orders.book_id = Books.book_id
WHERE order_id=?''', (order_id,))
order = cursor.fetchone()
cursor.execute('SELECT * FROM Customers')
customers = cursor.fetchall()
cursor.execute('SELECT * FROM Books')
books = cursor.fetchall()
return render_template('update_order.html', order=order, customers=customers,
```

```
books=books)
```

Видалити видавця

```
@app.route('/publishers/delete/<string:publisher_id>', methods=['POST'])
def delete_publisher(publisher_id):
conn = get_db()
cursor = conn.cursor()
cursor.execute('DELETE FROM Publisher WHERE publisher_id=?',
(publisher_id,))
conn.commit()
return redirect('/publishers')
```

Видалити книгу

```
@app.route('/books/delete/<int:book_id>', methods=['POST'])
def delete_book(book_id):
conn = get_db()
cursor = conn.cursor()
cursor.execute('DELETE FROM Books WHERE book_id=?', (book_id,))
conn.commit()
return redirect('/books')
```

Видалити замовлення

```
@app.route('/orders/delete/<int:order_id>', methods=['POST'])
def delete_order(order_id):
conn = get_db()
cursor = conn.cursor()
cursor.execute('DELETE FROM Orders WHERE order_id=?', (order_id,))
conn.commit()
return redirect('/orders')
```

Видалити клієнта

```
@app.route('/customers/delete/<int:customer_id>', methods=['POST'])
def delete_customer(customer_id):
conn = get_db()
```

```
cursor = conn.cursor()
cursor.execute('DELETE FROM Customers WHERE customer_id=?',
(customer_id,))
conn.commit()
return redirect('/customers')
```

Відобразити деталі видавця

```
@app.route('/publishers/<string:publisher_id>')
def display_publisher_details(publisher_id):
conn = get_db()
cursor = conn.cursor()
cursor.execute('SELECT * FROM Publisher WHERE publisher_id=?',
(publisher_id,))
publisher = cursor.fetchone()
if publisher:
return render_template('publisher_details.html', publisher=publisher)
else:
return "Publisher not found.", 404
```

Запуск програми з будь-якого адресу на порту 8080

```
if __name__ == "__main__":
app.run(host="0.0.0.0", port=8080)
```

3.2 Демонстрація готового веб-додатоку

На головній сторінці веб-додатку налічується 5 кнопок: книги, клієнти, замовлення, видавці та чат-бот-консультант (рис 3.1).

У якості чат-бота-консультанта був інтегрований штучний інтелект AssistantAPI, що спроможний консультувати на питання, проаналізувавши інформацію про книги та загальну інформацію о бібліотеці, яку я до нього завантажив. Чат-бот-консультант може давати контактну інформацію о бібліотеці, давати абсолютно любі поради щодо наявних книг, або, наприклад, яка книга найдорожча або найдешевша, які маються книги по певній темі, або зовсім порекомендувати навмання вибрану книгу (рис. 3.2, 3.3, 3.4).

Веб-додаток-бібліотека дозволяє створювати, продивлятись, редагувати та видаляти інформацію в сторінках книги, клієнти, замовлення та видавці (рис. 3.5, 3.6, 3.7, 3.8).

Рис. 3.1 Головна сторінка

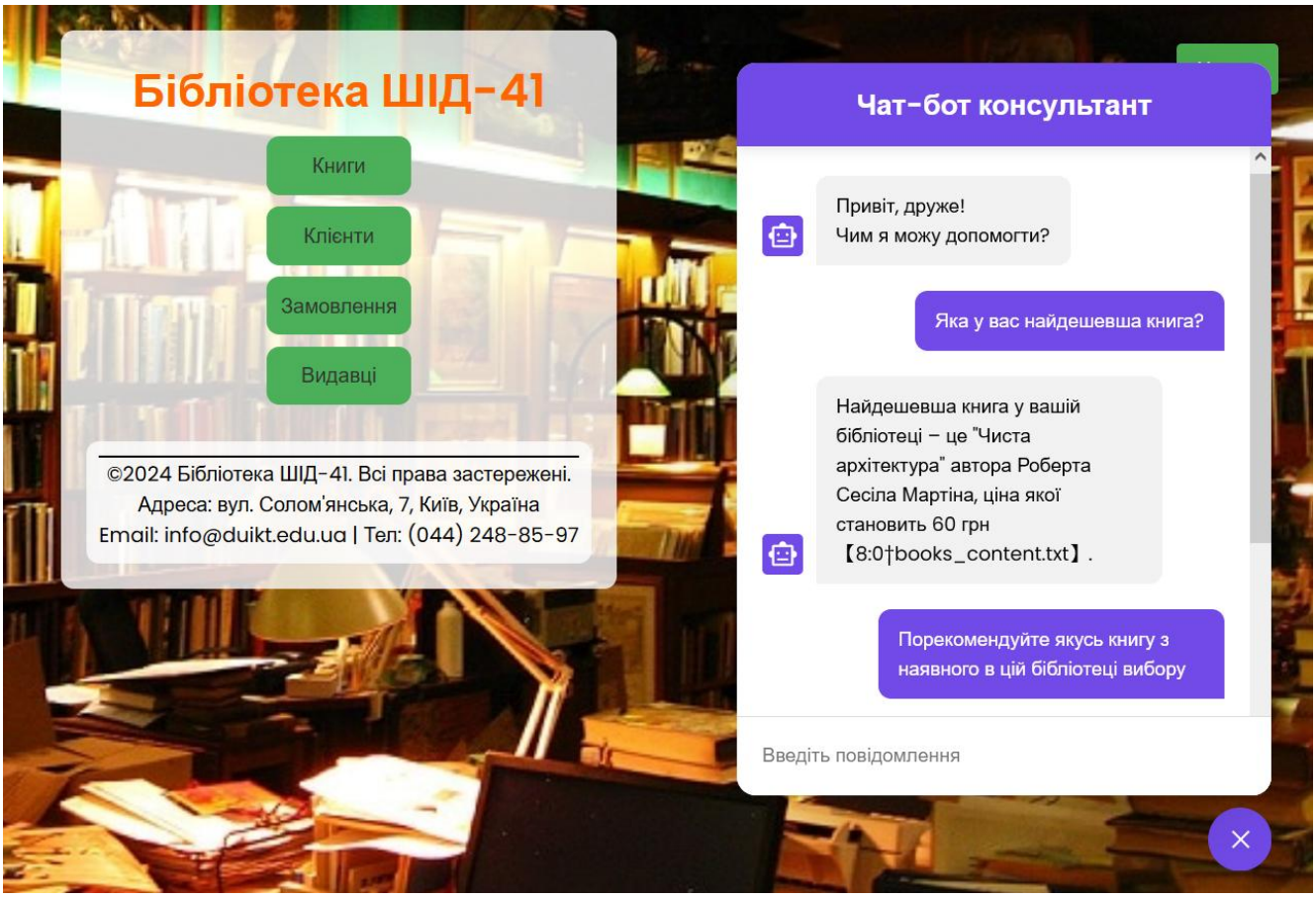


Рис. 3.2 Чат-бот-консультант

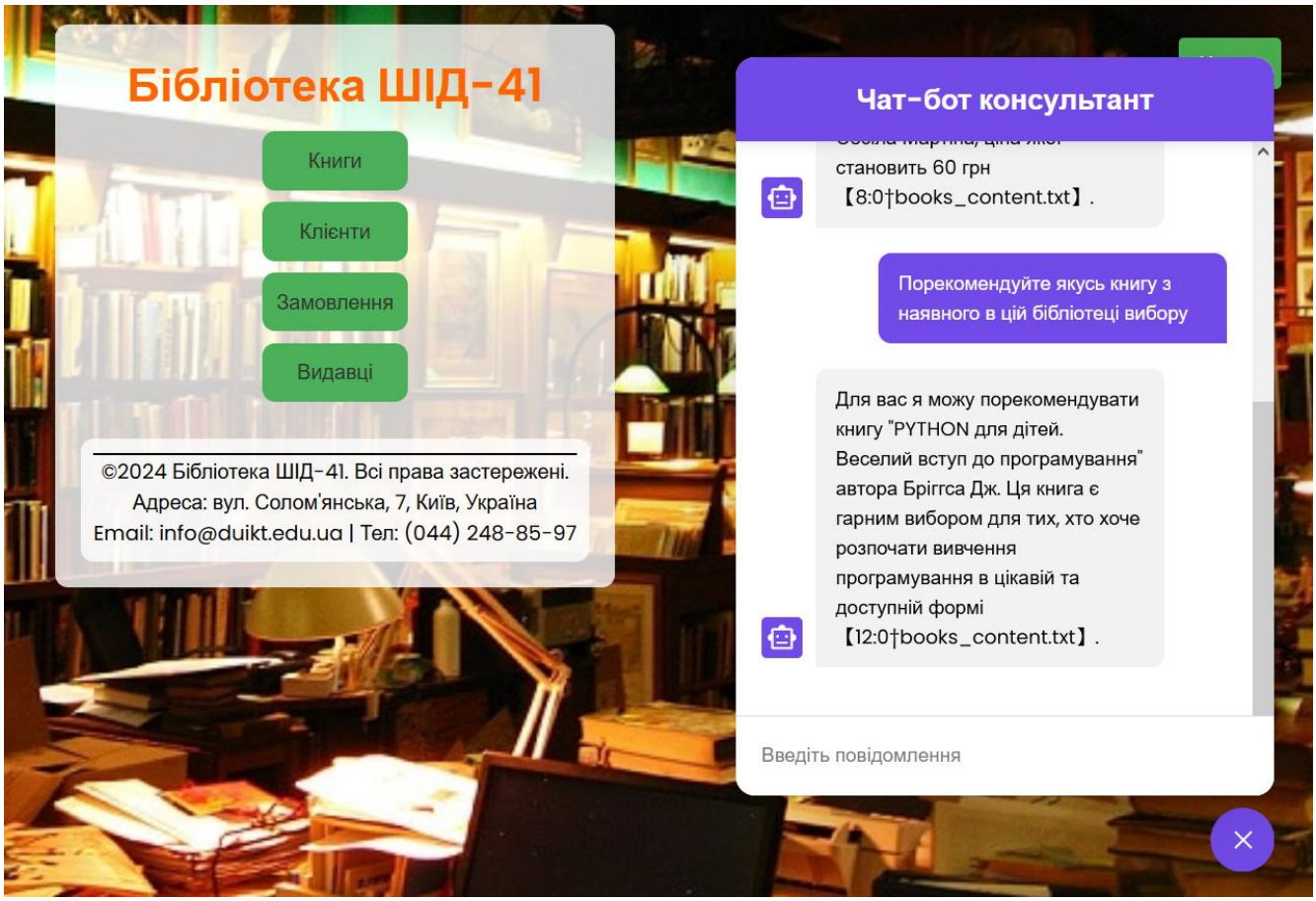


Рис. 3.3 Чат-бот-консультант

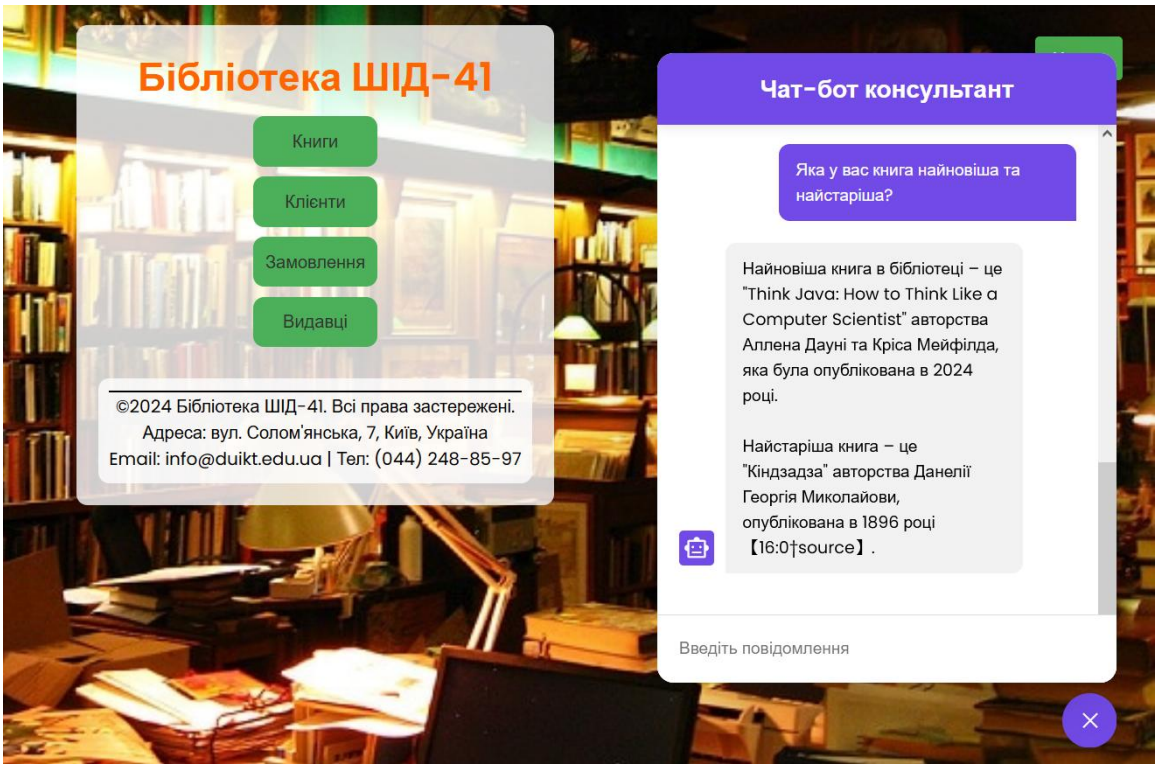
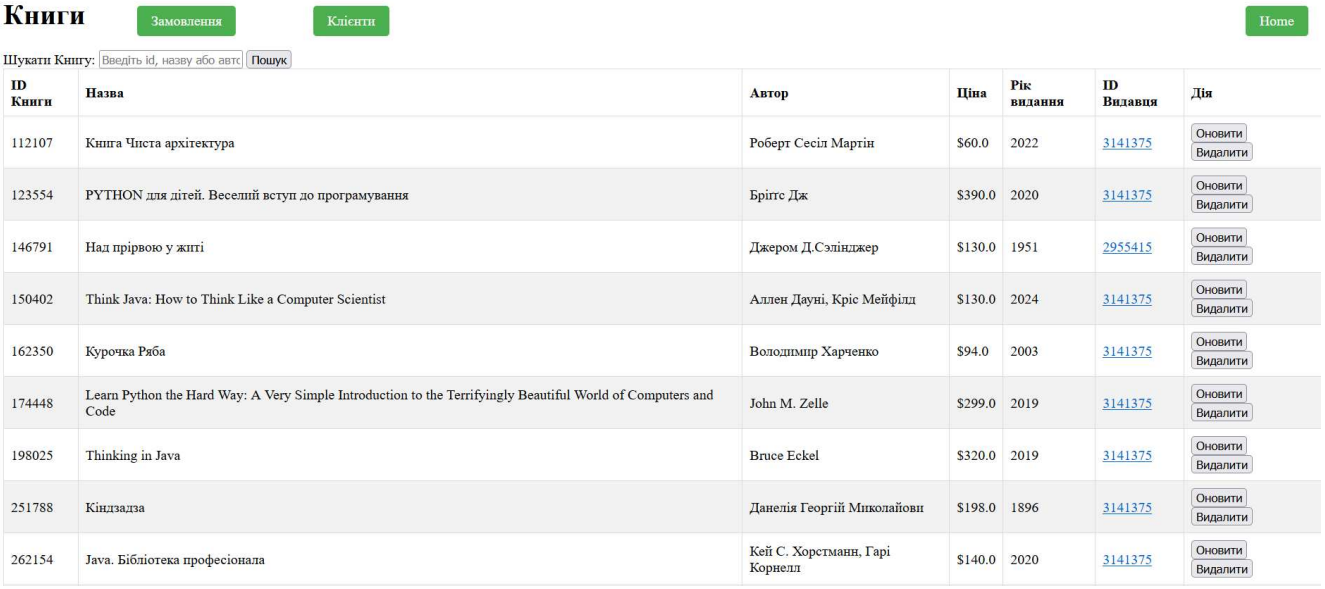


Рис. 3.4 Чат-бот-консультант

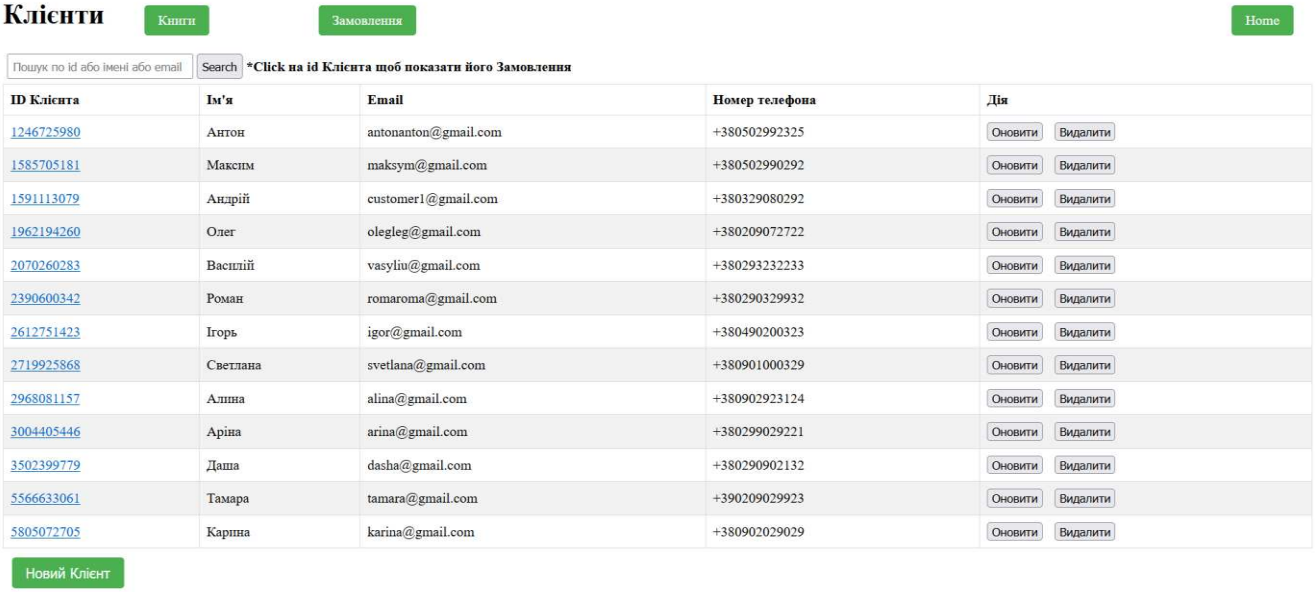


Книги Замовлення Клієнти Home

Шукати Книгу: Введіть id, назву або авто Пошук

ID Книги	Назва	Автор	Ціна	Рік видання	ID Видавця	Дія
112107	Книга Чиста архітектура	Роберт Сесіл Мартін	$60.0	2022	3141375	Оновити Видалити
123554	PYTHON для дітей. Веселий вступ до програмування	Брітгс Дж	$390.0	2020	3141375	Оновити Видалити
146791	Над прірвою у житі	Джером Д.Сэлинджер	$130.0	1951	2955415	Оновити Видалити
150402	Think Java: How to Think Like a Computer Scientist	Аллен Дауні, Кріс Мейфілд	$130.0	2024	3141375	Оновити Видалити
162350	Курочка Ряба	Володимир Харченко	$94.0	2003	3141375	Оновити Видалити
174448	Learn Python the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code	John M. Zelle	$299.0	2019	3141375	Оновити Видалити
198025	Thinking in Java	Bruce Eckel	$320.0	2019	3141375	Оновити Видалити
251788	Кіндзадза	Данелія Георгій Миколайови	$198.0	1896	3141375	Оновити Видалити
262154	Java. Бібліотека професіонала	Кей С. Хорстманн, Гарі Корнелл	$140.0	2020	3141375	Оновити Видалити

Рис. 3.5 Сторінка “книги”



Клієнти Книги Замовлення Home

Пошук по id або імені або email Search ***Click на id Клієнта щоб показати його Замовлення**

ID Клієнта	Ім'я	Email	Номер телефона	Дія
1246725980	Антон	antonanton@gmail.com	+380502992325	Оновити Видалити
1585705181	Максим	maksym@gmail.com	+380502990292	Оновити Видалити
1591113079	Андрій	customer1@gmail.com	+380329080292	Оновити Видалити
1962194260	Олег	olegleg@gmail.com	+380209072722	Оновити Видалити
2070260283	Василій	vasyliu@gmail.com	+380293232233	Оновити Видалити
2390600342	Роман	romaroma@gmail.com	+380290329932	Оновити Видалити
2612751423	Ігорь	igor@gmail.com	+380490200323	Оновити Видалити
2719925868	Светлана	svetlana@gmail.com	+380901000329	Оновити Видалити
2968081157	Алина	alina@gmail.com	+380902923124	Оновити Видалити
3004405446	Аріна	arina@gmail.com	+380299029221	Оновити Видалити
3502399779	Даша	dasha@gmail.com	+380290902132	Оновити Видалити
5566633061	Тамара	tamara@gmail.com	+390209029923	Оновити Видалити
5805072705	Карина	karina@gmail.com	+380902029029	Оновити Видалити

Новий Клієнт

Рис. 3.6 Сторінка “клієнти”

Замовлення

Книги Клієнти Home

ID Замовлення	ID Клієнта	Ім'я Клієнта	Назва Книги	Кількість	Дата Замовлення	Сума	Дія
12352912	1585705181	Максим	Дванадцять стільців	2	2024-05-11	$200	Оновити Видалити
12581106	2390600342	Роман	Курочка Ряба	1	2024-05-03	$94	Оновити Видалити
23713940	1962194260	Олег	Книга Пришвидшений курс Python. Практичний, проєктно-орієнтований вступ до програмування	2	2024-05-15	$1000	Оновити Видалити
27008864	1591113079	Андрій	Дванадцять стільців	7	2024-05-16	$700	Оновити Видалити
28292803	2390600342	Роман	Learn Python the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code	1	2024-05-05	$299	Оновити Видалити
29429877	1591113079	Андрій	Кіндзадза	1	2024-05-23	$156	Оновити Видалити
32743168	1962194260	Олег	Грокаємо алгоритми. Ілюстрований посібник для програмістів і допитливих	1	2024-05-03	$420	Оновити Видалити
44385718	1962194260	Олег	Think Java: How to Think Like a Computer Scientist	1	2024-05-11	$130	Оновити Видалити
72163825	1591113079	Андрій	Java For Dummies	1	2024-05-02	$230	Оновити Видалити
89333470	1591113079	Андрій	Кіндзадза	2	2023-07-26	$312	Оновити Видалити

Рис. 3.7 Сторінка “замовлення”

Видавці

Home

Пошук видавця: Введіть назву Видавця аб Пошук

ID Видавця	Назва Видавця	Адреса	Номер телефона	Website	Дія
3141375	Анатолій	gjnfn	+380505280272	https://www.google.com/search?client=firefox-b-d&q=types23	Оновити Видалити
2955415	Сергій	adsds	+380322020322	https://www.google.com/search?client=firefox-b-d&q=types+o123321bases	Оновити Видалити
3680603	Дарій	dasasdd	+380802923220	https://reactdom.com/flask/	Оновити Видалити
3136389	Олексій	dasfsf	+380299023922	https://www.google.com/search?client=firefox	Оновити Видалити
4558652	Маша	ggassdd	+380902900233	https://www.google.com/s324121344efox	Оновити Видалити
1376247	Таня	gasdasd	+380322023222	https://www.google.com/sear31215client=firefox	Оновити Видалити
2622373	Богдан	sdafsfa	+380322023111	https://www.google.com/search?client=fir32efo232	Оновити Видалити
2567264	Владислав	fasddsa	+380902900235	https://www.google.com/search23client=fir42efox	Оновити Видалити
2782222	Артем	dasdsasf	+380322020567	https://www.google.com/search?client=3211efox	Оновити Видалити

Додати Видавця

Рис. 3.8 Сторінка “видавці”

ВИСНОВКИ

В рамках цього проекту був розроблений та впроваджений веб-додаток для генерації тематичних повідомлень з використанням генеративних моделей штучного інтелекту. Ідея проекту полягала у створені чат-бота-консультанта для веб-додатку-бібліотеки, який здатен тренеруватись на вхідний інформації та дотримуватися впроваджених інструкцій, що значно допоможе підвищити ефективність комунікацій між клієнтами та компаніями.

Як результат, функціонал бібліотеки дозволяє продивлятись, створювати, редагувати та видаляти книги, клієнтів, видавців та замовлення, тоді як чат-бот-консультант здатен відповідати на будь-які запити користувачів, стосовні наявних книг або надавати загальну інформацію о компанії.

На початковому етапі було проаналізовано предметну область, а саме: вивчено поняття інтелектуальних мовних моделей та проведено огляд популярних веб-сайтів з інтегрованими чат-ботами на основі штучного інтелекту. Наступним кроком стало обрання засобів реалізації проекту: огляд існуючих засобів створення веб-додатку та інтеграції чат-бота-консультанта.

Розроблений веб-додаток успішно пройшов тестування і повністю задовольняє поставлені вимоги. Він готовий до впровадження та використання в реальних умовах.

Завдяки виконанню цього проекту було отримано цінні знання та практичний досвід у сферах розробки веб-додатків, інтеграції інтелектуальних моделей, а також проектування зручних користувацьких інтерфейсів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Technologies I. What are web frameworks and why you need them?. Medium. URL: https://intelegain-technologies.medium.com/what-are-web-frameworks-and-why-you-need-them-c4e8806bd0fb (date of access: 30.05.2024).
2. Web Accessibility: Web Standards and Regulatory Compliance / P. H. Lauke et al. Springer London, Limited, 2006. 248 с.
3. Django introduction - Learn web development | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction (date of access: 30.05.2024).
4. Tushar Sawant Afroj Satwilkar Vaibhav Shirke Prof Santosh V. Jadhav. Survey on Django Based Web Application to Empower Skilled People. International Journal of Innovative Research in Science. 2021. P. 4999. URL: https://www.ijirset.com/upload/2021/may/146_survey%20paper%20ESP_NC.pdf (date of access: 15.05.2024).
5. Joshi B. Beginning JQuery 2 for ASP. NET Developers: Using JQuery 2 with ASP. NET Web Forms and ASP. NET MVC. Springer, 2014.
6. Паздрій Ігор. Порівняльний аналіз систем розробки програмного забезпечення на основі фреймворків. Вісник Хмельницького національного університету. 2023. 155 с. URL: http://journals.khnu.km.ua/vestnik/wp-content/uploads/2023/03/vknu-ts-2023-n1317-155-161.pdf.
7. What is Flask Python - Python Tutorial. Learn Python Programming - Python Tutorial. URL: https://pythonbasics.org/what-is-flask-python/ (date of access: 30.05.2024).
8. FastAPI: Everything you need to know about the most widely used Python web framework for Machine Learning. Data Science Courses | DataScientest. URL: https://datascientest.com/en/fastapi-everything-you-need-to-know-about-the-most-widely-used-python-web-framework-for-machine-learning (date of access: 15.05.2024).
9. What is Django Web Framework? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-django-web-framework/ (date of access: 30.05.2024).

10. Осмоловська Ю. С. Веб-додаток реалізований мовою програмування Ruby та фреймворком Ruby on Rails. м. Суми, р. 2021. 13 с.
11. Araujo M. Exploring the Python Pyramid Framework: Building Web Applications with Power and Flexibility. Medium. URL: https://medium.com/@matheushcamilo1997/exploring-the-python-pyramid-framework-building-web-applications-with-power-and-flexibility-2594a97e6237 (date of access: 15.05.2024).
12. What is Angular in Web Development? - Framework Overview - Glossary. Sanity.io. URL: https://www.sanity.io/glossary/angular (дата звернення: 15.05.2024).
13. What Is Vue JS?. Built In. URL: https://builtin.com/software-engineering-perspectives/vue-js (дата звернення: 15.05.2024).
14. React Introduction - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/reactjs-introduction/ (date of access: 15.05.2024).
15. What is Svelte?. Educative. URL: https://www.educative.io/answers/what-is-svelte (дата звернення: 15.05.2024).
16. S R. A. What is SQLite? And When to Use It?. Simplilearn.com. URL: https://www.simplilearn.com/tutorials/sql-tutorial/what-is-sqlite (дата звернення: 15.05.2024).
17. What is PostgreSQL? – Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/rds/postgresql/what-is-postgresql/ (дата звернення: 15.05.2024).
18. What is MySQL? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-mysql/ (date of access: 15.05.2024).

Апробація результатів та публікації

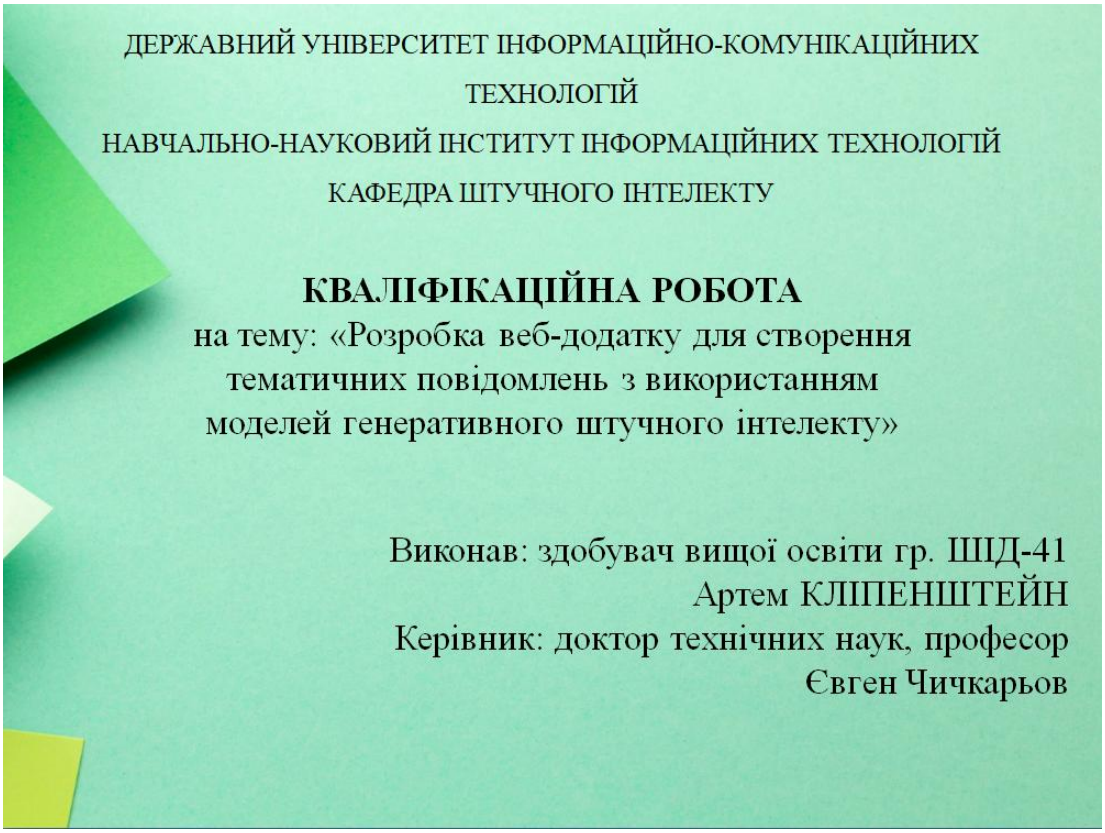
1. EXPLORING THE IMPLEMENTATION OF INFORMATION AND COMMUNICATIONS TECHNOLOGY (ICT) IN MUNICIPAL AUTHORITIES, 24 квітня 2024 року, кафедра Інженерії програмного забезпечення Навчальнонаукового інституту інформаційних технологій Державного університету інформаційно-комунікаційних технологій, м. Київ., p_2661_45497999.pdf (duikt.edu.ua).

2. EXPLORING CLOUD SERVICES TECHNOLOGY, 24 квітня 2024 року, кафедра Інженерії програмного забезпечення Навчальнонаукового інституту інформаційних технологій Державного університету інформаційно-комунікаційних технологій, м. Київ., p_2661_45497999.pdf (duikt.edu.ua).

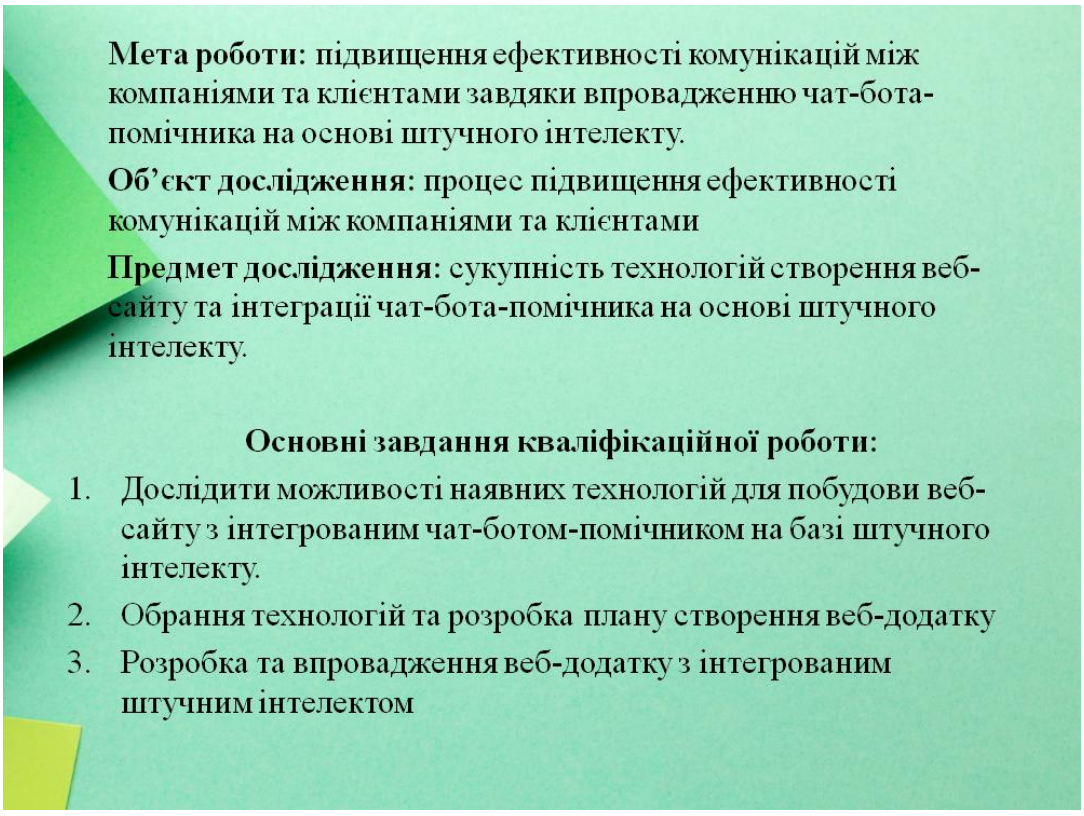
3. EXPLORING THE IMPLEMENTATION OF ARTIFICIALINTELLIGENCE IN MANUFACTURING, 15 грудня 2023р., кафедра Комп'ютерних наук Навчально-наукового інституту інформаційних технологій Державного університету телекомунікацій, м. Київ., p_2626_86233288.pdf (duikt.edu.ua).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

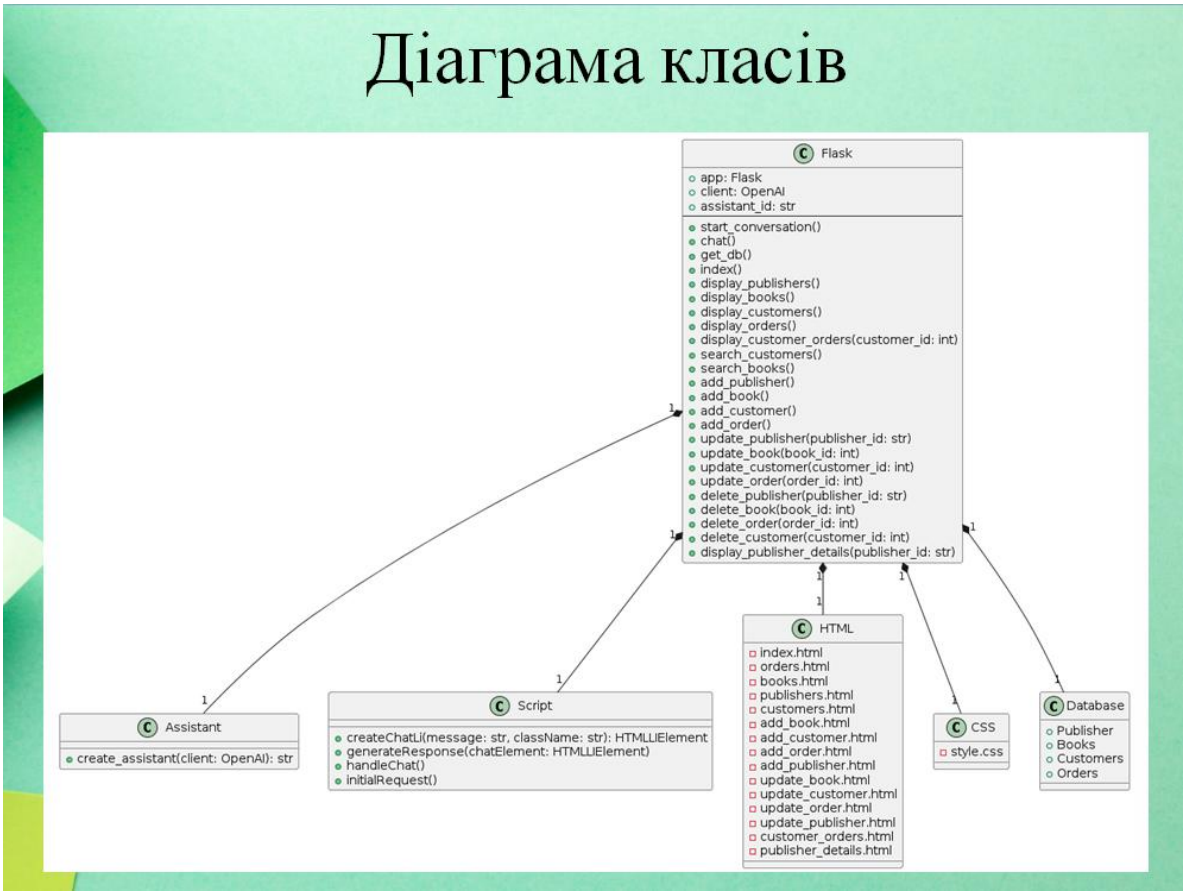
Слайд 1



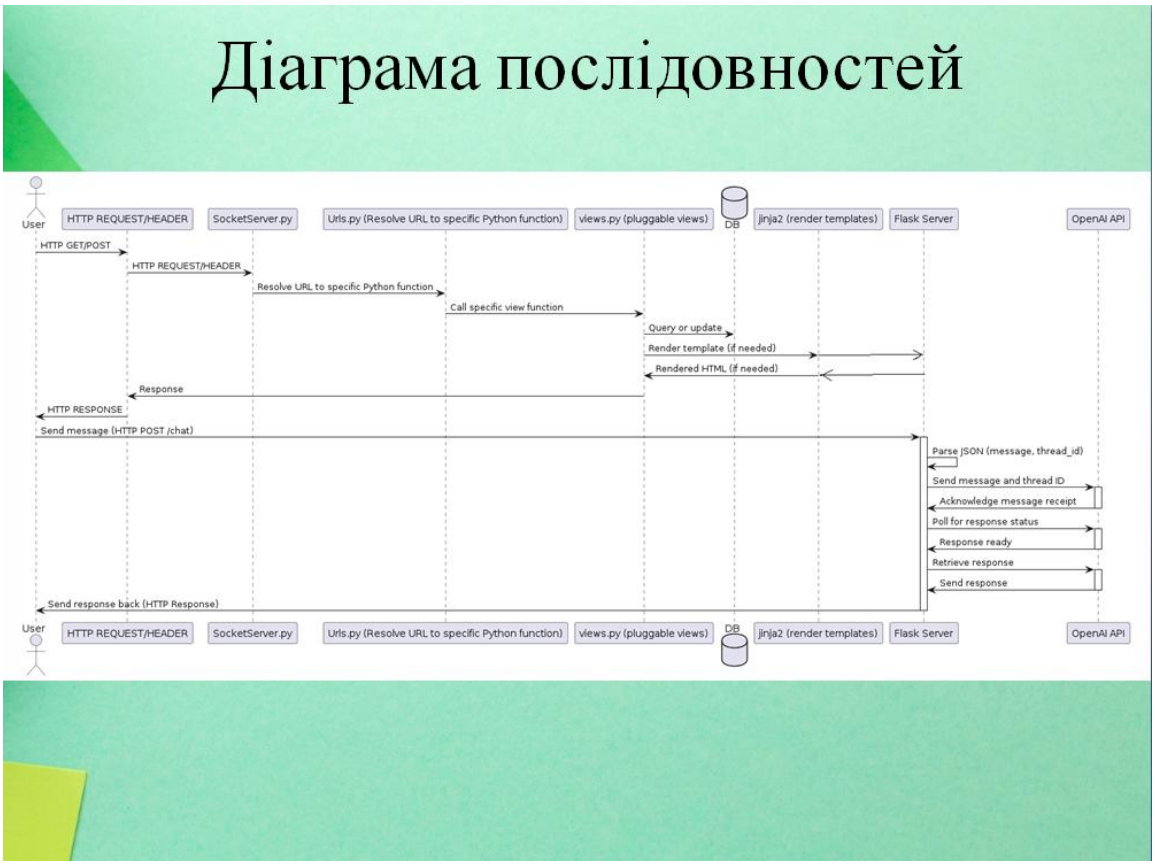
Слайд 2



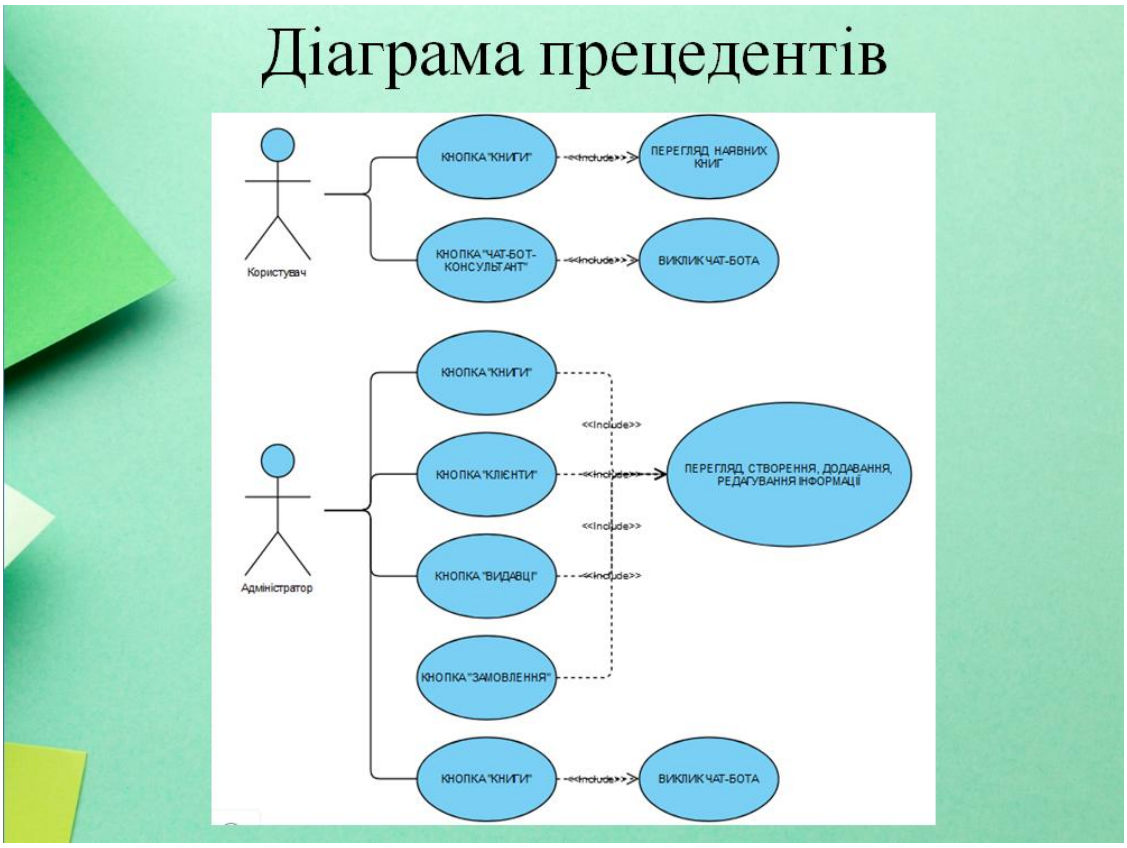
Слайд 3



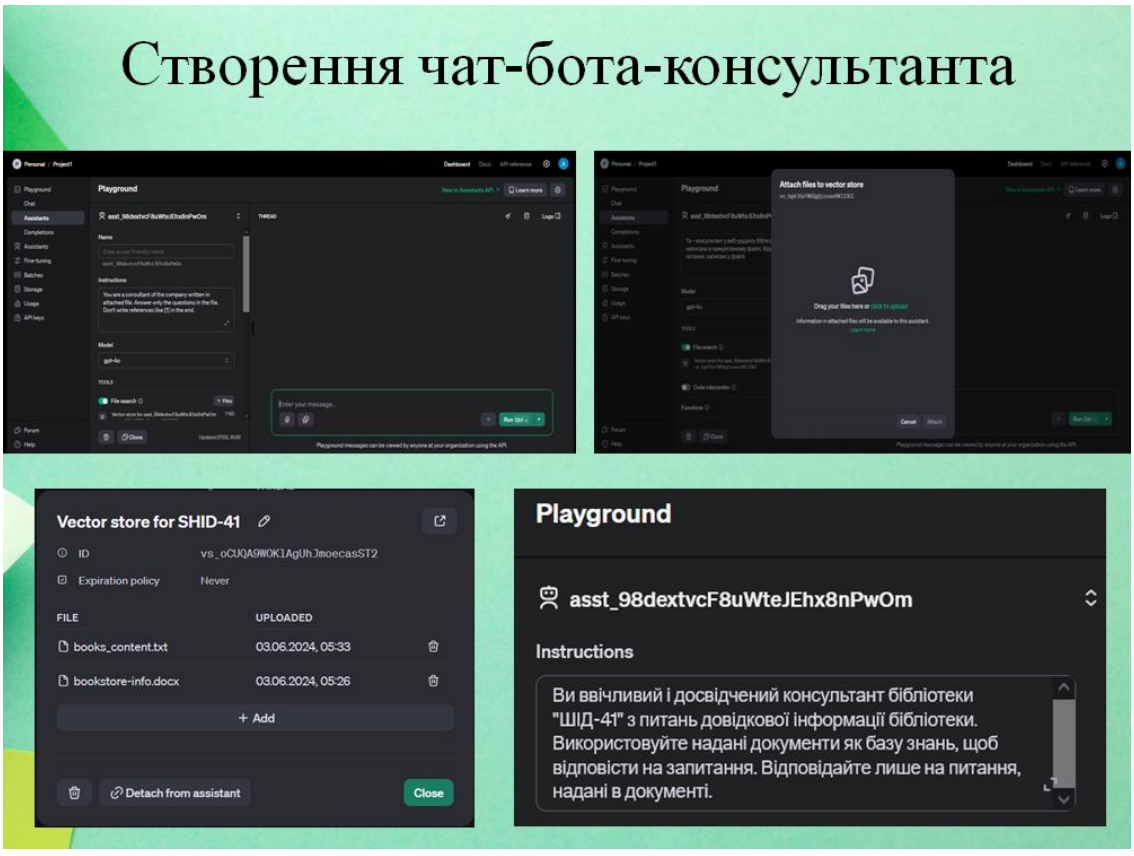
Слайд 4



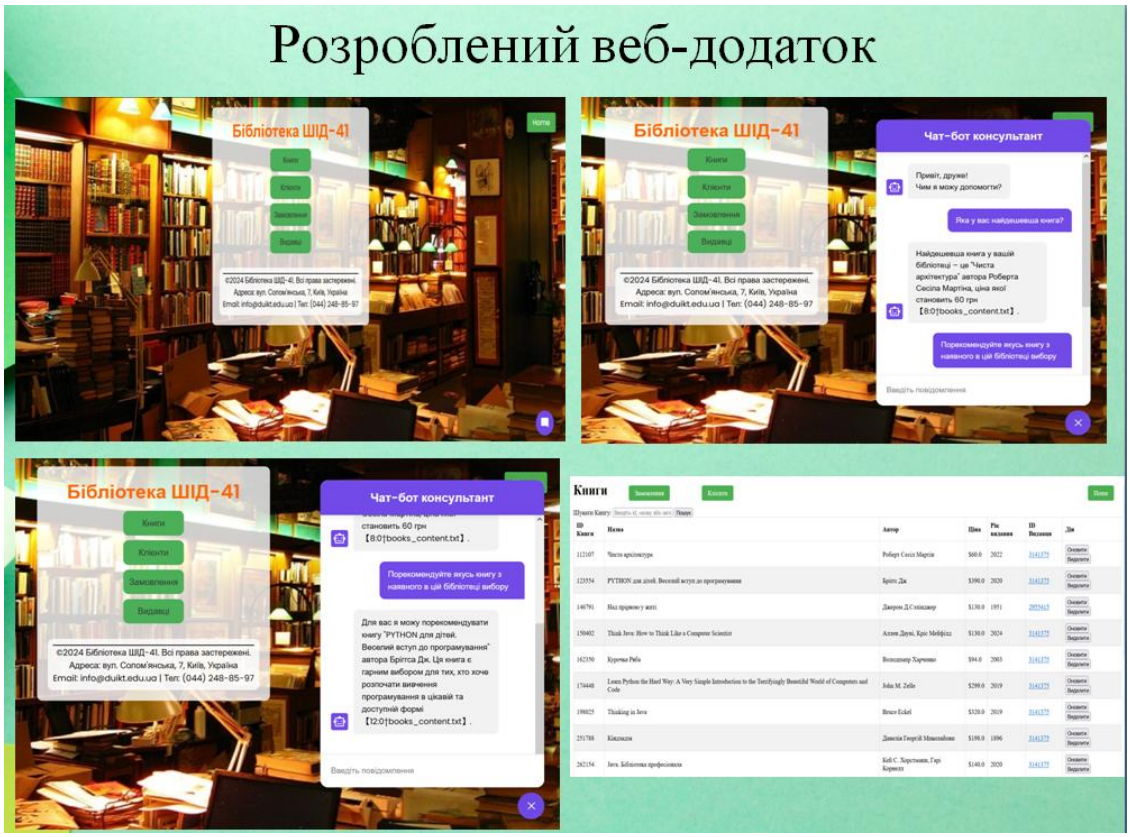
Слайд 5



Слайд 6



Слайд 7



Слайд 8

Висновки

В рамках цього проекту був розроблений та впроваджений веб-додаток для генерації тематичних повідомлень з використанням генеративних моделей штучного інтелекту. Ідея проекту полягала у створені чат-бота-консультанта для веб-додатку-бібліотеки, який здатен тренеруватись на вхідний інформації та дотримуватися впроваджених інструкцій, що значно допоможе підвищити ефективність комунікацій між клієнтами та компаніями.

Як результат, функціонал бібліотеки дозволяє продивлятись, створювати, редагувати та видаляти книги, клієнтів, видавців та замовлення, тоді як чат-бот-консультант здатен відповідати на будь-які запити користувачів, стосовні наявних книг або надавати загальну інформацію о компанії.

На початковому етапі було проаналізовано предметну область, а саме: вивчено поняття інтелектуальних мовних моделей та проведено огляд популярних веб-сайтів з інтегрованими чат-ботами на основі штучного інтелекту. Наступним кроком стало обрання засобів реалізації проекту: огляд існуючих засобів створення веб-додатку та інтеграції чат-бота-консультанта.

Розроблений веб-додаток успішно пройшов тестування і повністю задовольняє поставлені вимоги. Він готовий до впровадження та використання в реальних умовах.

Завдяки виконанню цього проекту було отримано цінні знання та практичний досвід у сферах розробки веб-додатків, інтеграції інтелектуальних моделей, а також проектування зручних користувацьких інтерфейсів.