Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Центр ННЦЗФН
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Інтернет-магазин із чат-ботом на основі машинного навчання
(тема)

Виконала:
здобувачка четвертого року навчання,
групи ІТШІз-21-1
Шумілова Ольга
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ас. Марія Погурська
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Центр ННЦЗФН

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачці Шуміловій Ользі Станіславівні
(прізвище, ім'я, по батькові)

1. Тема роботи Інтернет-магазин із чат-ботом на основі машинного навчання

затверджена наказом університету від 7 травня 20 25 р. № 80Стз
2. Термін подання студентом роботи до екзаменаційної комісії 24 червня 20 25 р.
3. Вихідні дані до роботи Наукові статті та публікації, матеріали офіційної документації React, Node.js, MongoDB та Python, Аналітичні дані та кейси використання чат-ботів в електронній комерції, відкриті набори даних для навчання моделей чат-ботів, Інтернет-ресурси, навчальні курси та форуми розробників

4. Перелік питань, що потрібно опрацювати в роботі
1) Теоретичні основи побудови інтелектуальних систем у сфері електронної комерції
2) Постановка задачі та обґрунтування технологій
3) Проєктування та реалізація інтелектуальної системи
4) Тестування та перевірка системи

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	05.05.2025	виконано
2	Аналіз предметної галузі	07.05.2025	виконано
3	Огляд існуючих рішень інтернет-магазинів і чат-	10.05.2025	виконано
	ботів		
4	Вивчення технологій машинного навчання для	16.05.2025	виконано
	чат-ботів		
5	Розробка архітектури інтернет-магазину з чат-	21.05.2025	виконано
	ботом		
6	Реалізація чат-бота на основі машинного	30.05.2025	виконано
	навчання		
7	Тестування та налагодження системи	07.06.2025	виконано
8	Написання пояснювальної записки	11.06.2025	виконано
9	Нормоконтроль	15.06.2025	виконано
10	Підготовка презентації та доповіді	17.06.2025	виконано
11	Попередній захист	20.06.2025	виконано
12	Рецензування	22.06.2025	виконано
13	Захист перед ЕК	24.06.2025	

Дата видачі завдання 31 березня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Марія Погурська
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 93 с., 22 рис., 2 табл., 5 дод., 21 джерело.

ВЕКТОРНИЙ ПОШУК, ІНТЕРНЕТ-МАГАЗИН, МАШИННЕ НАВЧАННЯ, ЧАТ-БОТ, FLASK, MONGODB, NLP, PYTORCH, REACT.

Об’єкт дослідження – процес цифрової трансформації торгівлі та застосування інтелектуальних систем для покращення взаємодії з користувачем у сфері електронної комерції.

Предмет дослідження – розробка веб-застосунку інтернет-магазину з інтегрованим чат-ботом на основі машинного навчання, що автоматизує обробку користувацьких запитів.

Мета роботи – створення інноваційної системи інтернет-магазину з підтримкою чат-бота, який інтегрує інструменти штучного інтелекту для обробки природної мови та пошуку за зображеннями.

Методи дослідження – теоретичний аналіз сучасних технологій веб-розробки, машинного навчання та NLP, практична реалізація клієнтської та серверної частин, використання алгоритмів обробки природної мови на основі інтенцій (намір) та великих мовних моделей (LLM), розробка системи векторного пошуку, функціональне тестування.

У результаті роботи інтегровано сучасні методи машинного навчання для обробки текстових і візуальних запитів. Реалізовано мультимодальний інтерфейс для пошуку товарів за текстом і зображеннями. Впроваджено гібридну модель чат-бота на основі інтенційної логіки та генеративної LLM. Проведено тестування, яке показало точність розпізнавання намірів користувача понад 92% та релевантність пошуку за зображеннями понад 50% за косинусною метрикою. Результати підтверджують інноваційність і конкурентоспроможність системи на національному рівні.

ABSTRACT

Bachelor’s thesis contains: 93 pp., 22 fig., 2 tabl., 5 ann., 21 references.

CHATBOT, FLASK, MACHINE LEARNING, MONGODB, NLP, ONLINE STORE, PYTORCH, REACT, VECTOR SEARCH.

Object of research – the process of digital transformation in commerce and the application of intelligent systems to improve user interaction in e-commerce.

Subject of research – development of a web application for an online store with an integrated machine learning-based chatbot that automates the processing of user requests.

Purpose of the work – creation of an innovative online store system supporting a chatbot that integrates artificial intelligence tools for natural language processing and image-based search.

Research methods – theoretical analysis of modern web development technologies, machine learning, and NLP; practical implementation of client and server parts; use of natural language processing algorithms based on intents and large language models (LLM); development of a vector search system; functional testing.

As a result of the work, modern machine learning methods were integrated to process textual and visual user queries. A multimodal interface for searching products by text and images was implemented. A hybrid chatbot model based on intent logic and generative LLM was introduced. Testing showed intent recognition accuracy above 92% and image search relevance over 50% by cosine similarity metric. The results confirm the innovation and competitiveness of the system at the national level.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – інтерфейс прикладного програмування;

CLIP – Contrastive Language–Image Pre-training – модель для поєднання зображень і тексту;

CNN – Convolutional Neural Network – згорткова нейронна мережа, модель комп'ютерного зору;

CRM – Customer Relationship Management – система управління взаємовідносинами з клієнтами;

CRUD – Create, Read, Update, Delete – створення, читання, оновлення, видалення;

GPT – Generative Pre-trained Transformer – модель генерації текстів;

HTTP – HyperText Transfer Protocol – протокол передавання гіпертексту;

JSON – JavaScript Object Notation – нотація об'єктів JavaScript;

LLM – Large Language Model – велика мовна модель;

NLP – Natural Language Processing – обробка природної мови.

ВСТУП

У сучасних умовах цифрової трансформації торгівлі зростає потреба у впровадженні інтелектуальних систем для покращення взаємодії з користувачем у сфері електронної комерції. Особливо актуальним є застосування машинного навчання та обробки природної мови для автоматизації обробки користувацьких запитів, що сприяє підвищенню якості сервісу і конкурентоспроможності бізнесу.

Актуальність цієї роботи обумовлена розвитком технологій штучного інтелекту, які дозволяють створювати чат-ботів з більш гнучким і природним інтерфейсом, а також впроваджувати мультимодальний пошук товарів за текстом і зображеннями. Це дає змогу інтернет-магазинам підвищити рівень обслуговування та оптимізувати процеси пошуку. Застосування таких технологій особливо важливе у контексті зростаючих вимог користувачів до швидкості і якості онлайн-сервісів.

Метою роботи є розробка інноваційного веб-застосунку інтернет-магазину з чат-ботом на основі машинного навчання, який забезпечує ефективну обробку текстових і візуальних запитів користувачів. Результати можуть бути використані у сфері електронної комерції для кращого обслуговування клієнтів та оптимізації бізнес-процесів.

Історично, розвиток чат-ботів і технологій NLP пройшов значний шлях від простих правил і шаблонів до складних моделей з глибоким навчанням і великими мовними моделями (LLM). Використання таких підходів у поєднанні з векторним пошуком відкриває нові можливості для побудови інтелектуальних систем у комерційній сфері. Запровадження цих технологій дозволяє забезпечити більш точне розуміння намірів користувачів, адаптивність системи до нестандартних запитів, а також підвищує загальну якість взаємодії з клієнтом.

1 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ У СФЕРІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

1.1 Застосування штучного інтелекту в електронній комерції

1.1.1 Роль ШІ у трансформації онлайн-бізнесу

Штучний інтелект (ШІ) є однією з найважливіших технологій, що визначають майбутнє електронної комерції. Завдяки своїм можливостям обробки великих обсягів даних, автоматизації процесів, а також здатності до навчання та самовдосконалення, ШІ відкриває нові горизонти для розвитку онлайн-бізнесу, підвищуючи ефективність обслуговування клієнтів, збільшуючи продажі та поліпшуючи взаємодію з користувачами. У сучасних умовах, коли споживачі стали вимогливішими, а конкуренція на ринку онлайн-продажів є надзвичайно високою, впровадження ШІ у бізнес-процеси здатне значно підвищити конкурентоспроможність компаній.

Порівняно з іншими галузями, вигода від впровадження ШІ в електронній комерції є особливо вагомою, оскільки саме тут технологія має безпосередній вплив на всі ключові аспекти бізнесу – від персоналізації пропозицій і автоматизованого обслуговування клієнтів до оптимізації логістики та управління запасами. У той час як в інших сферах впровадження ШІ часто зосереджується на внутрішніх процесах або довготривалих дослідженнях, у сфері e-commerce (електронна комерція) результати проявляються майже миттєво й мають прямий вплив на доходи компанії. Це робить електронну комерцію однією з найперспективніших і найдинамічніших платформ для реалізації потенціалу штучного інтелекту. На рисунку 1.1 вказано, що вигода від впровадження ШІ в електронній комерції на порядок вища ніж в інших галузях [1].

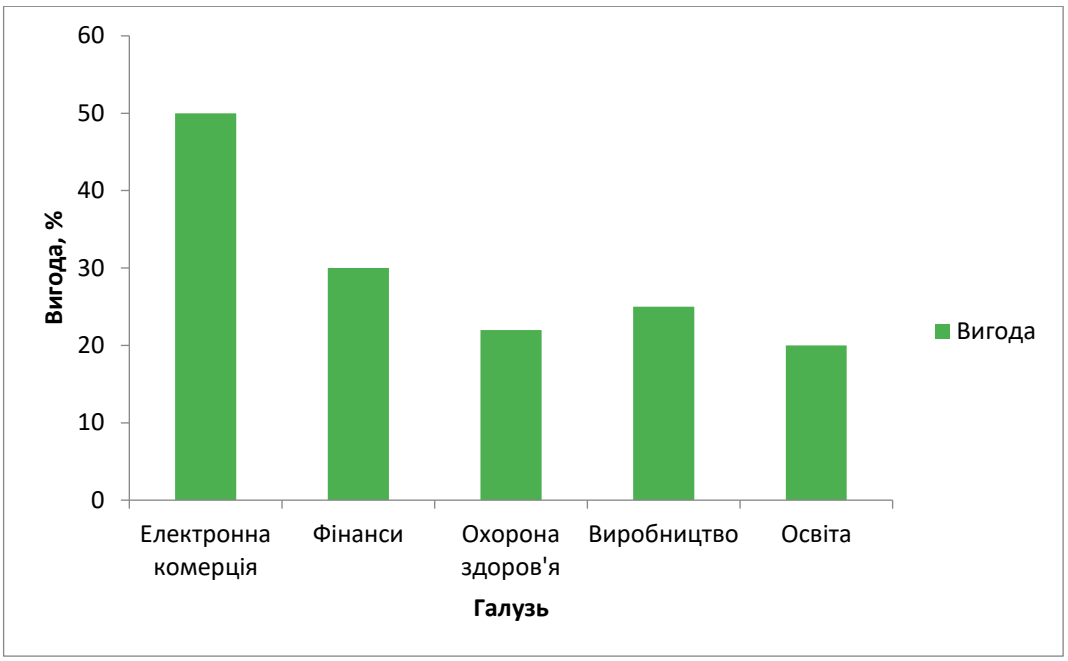

Рисунок 1.1 – Процент вигоди від впровадження ШІ за галузями

Таким чином, використання ШІ в електронній комерції не лише забезпечує оперативну вигоду, але й формує стратегічну перевагу на довгострокову перспективу. Постійний розвиток технологій штучного інтелекту відкриває нові можливості для бізнесу, що прагне адаптуватися до цифрової епохи та відповідати очікуванням сучасного споживача.

Однією з найбільш розповсюджених форм застосування ШІ в електронній комерції є автоматизація процесів обслуговування клієнтів. Чат-боти, що використовують технології машинного навчання та NLP (Natural Language Processing – обробка природної мови), дозволяють забезпечити негайну відповідь на запити користувачів. Вони здатні в режимі реального часу відповідати на питання щодо товарів, їх характеристик, наявності на складі, ціни, способів оплати та доставки, а також допомогти з оформленням замовлень.

Завдяки використанню ШІ чат-боти можуть навчатися на основі взаємодії з користувачами та вдосконалювати свої відповіді з часом. Більше того, за допомогою аналітики даних, бот може прогнозувати, які питання найімовірніше будуть поставлені, і заздалегідь надавати готові відповіді, що

підвищує ефективність обслуговування. Це дає можливість компаніям значно скоротити витрати на персонал та підвищити задоволеність клієнтів, адже клієнти отримують миттєві та точні відповіді без необхідності чекати на відповіді від живого оператора.

Крім того, чат-боти з підтримкою штучного інтелекту можуть інтегруватися з CRM (Customer Relationship Management – управління взаємовідносинами з клієнтами), що дозволяє персоналізувати обслуговування на основі історії покупок, попередніх запитів та поведінки користувача. Такий підхід сприяє формуванню індивідуального досвіду для кожного клієнта, підвищуючи рівень залучення та лояльності. У результаті, компанії можуть не лише оперативно вирішувати проблеми, а й проактивно пропонувати клієнтам релевантні товари чи послуги, що сприяє зростанню продажів.

Sephora має 2 бота для Facebook. Перший чат-бот створено постачальником Assi.st. «Sephora Reservation Assistant» – це бот для бронювання зустрічей, щоб зарезервувати макіяж у магазинах Sephora. Бот обробляє природну мову. Коефіцієнт конверсії бота Sephora на 11% вищий порівняно з будь-яким іншим каналом для бронювання зустрічей з оновленим дизайном у магазині.

Другий чат-бот створено постачальником ModiFace. «Color Match» для «Sephora Virtual Artist» – це бот для підбору відтінків. Клієнт сканує зображення, об'єкт або обличчя знаменитості та переглядає список відповідних помад. Бот використовує технології комп’ютерного зору та машинного навчання для точного розпізнавання кольорів і їх зіставлення з продукцією бренду. Це значно спрощує процес вибору косметики, роблячи його персоналізованим та інтерактивним.

На рисунку 1.2 показано роботу двох чат-ботів Sephora для Facebook: «Sephora Reservation Assistant» для бронювання макіяжу у магазинах і «Color Match» для підбору відтінків помад за допомогою сканування зображень.

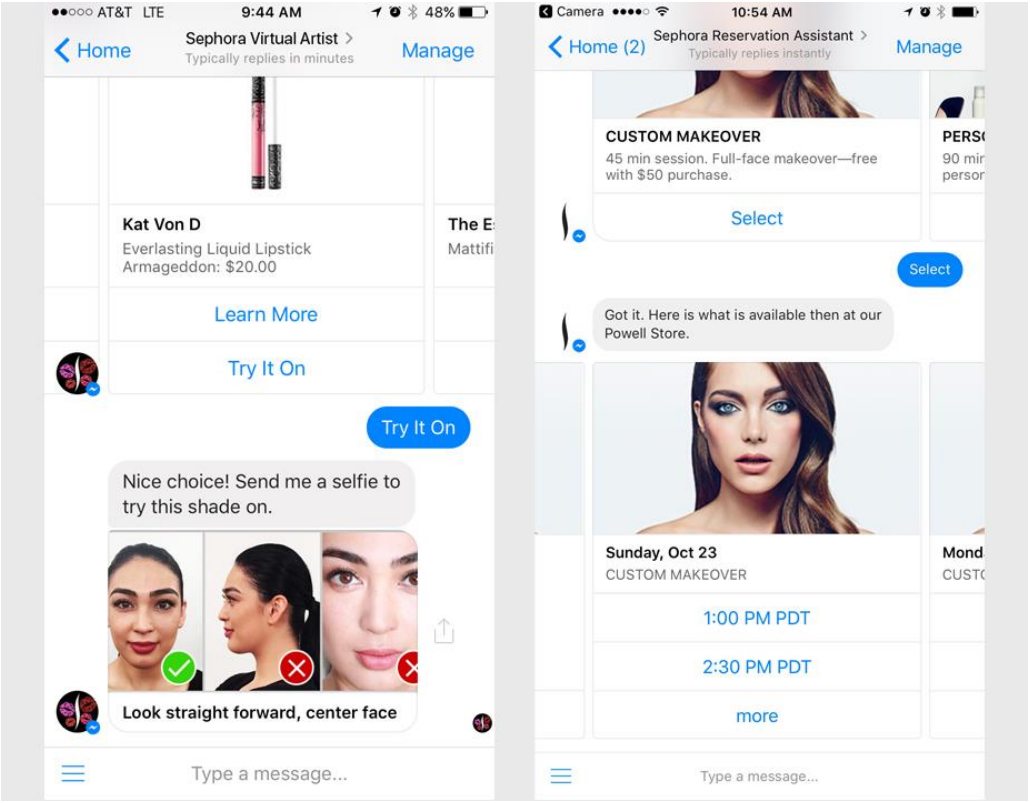

Рисунок 1.2 – Демонстрація роботи чат-ботів Sephora для Facebook

Отже, використання чат-ботів у електронній комерції є не лише прикладом ефективної автоматизації, а й важливим інструментом підвищення якості взаємодії з клієнтами. ШІ-технології дозволяють не просто замінити людину в типових запитах, а створити повноцінну систему підтримки, яка адаптується до потреб користувача. Подальше вдосконалення таких рішень сприятиме ще більшій персоналізації обслуговування та підвищенню лояльності клієнтів, що є критично важливим фактором успіху в умовах сучасної конкуренції на ринку e-commerce.

1.1.2 Персоналізація пропозицій та рекомендацій

Ще однією важливою сферою застосування ШІ є персоналізація обслуговування клієнтів. Завдяки здатності ШІ аналізувати поведінку

користувачів на веб-сайті або в мобільних додатках, можна створювати персоналізовані пропозиції, що відповідають інтересам і потребам кожного клієнта. Персоналізація не тільки покращує досвід користувача, але й значно збільшує ймовірність покупки.

Для цього ШІ аналізує дані про попередні покупки користувачів, їх перегляди товарів, вподобання та оцінки. Такі системи, як рекомендаційні системи, дозволяють пропонувати найбільш релевантні товари, що можуть зацікавити конкретного користувача. Наприклад, найбільші онлайн-платформи, такі як Amazon чи Netflix, активно використовують ШІ для формування персоналізованих рекомендацій, які значно покращують досвід користувачів і стимулюють їх до частіших покупок або переглядів.

Використання персоналізованих рекомендацій дозволяє компаніям не лише збільшити обсяги продажу, а й побудувати довгострокові відносини з клієнтами, що є важливим аспектом в умовах високої конкуренції на ринку. Наприклад, глобальний магазин косметики Sephora також має чат-бота, який працює на платформі обміну повідомленнями Kik (месенджер). Він пропонує користувачеві пройти короткий тест, а потім задає цільові запитання про смаки клієнтів. Потім пропонує рекомендації щодо продуктів. Інтерактивний та цікавий, він дозволяє здійснювати покупки, не виходячи з месенджера. Такий динамічний підхід забезпечує ще глибший рівень індивідуалізації, дозволяючи брендам пропонувати максимально релевантний контент, знижки або акції саме тоді, коли це найбільш доречно. Це не тільки сприяє підвищенню конверсій, а й створює відчуття турботи та уваги до потреб клієнта.

Рисунок 1.3 ілюструє роботу чат-бота Sephora на платформі обміну повідомленнями Kik. Бот проводить тестування користувача, задає питання про смаки та пропонує персоналізовані рекомендації продуктів, що дозволяє здійснювати покупки безпосередньо в месенджері.

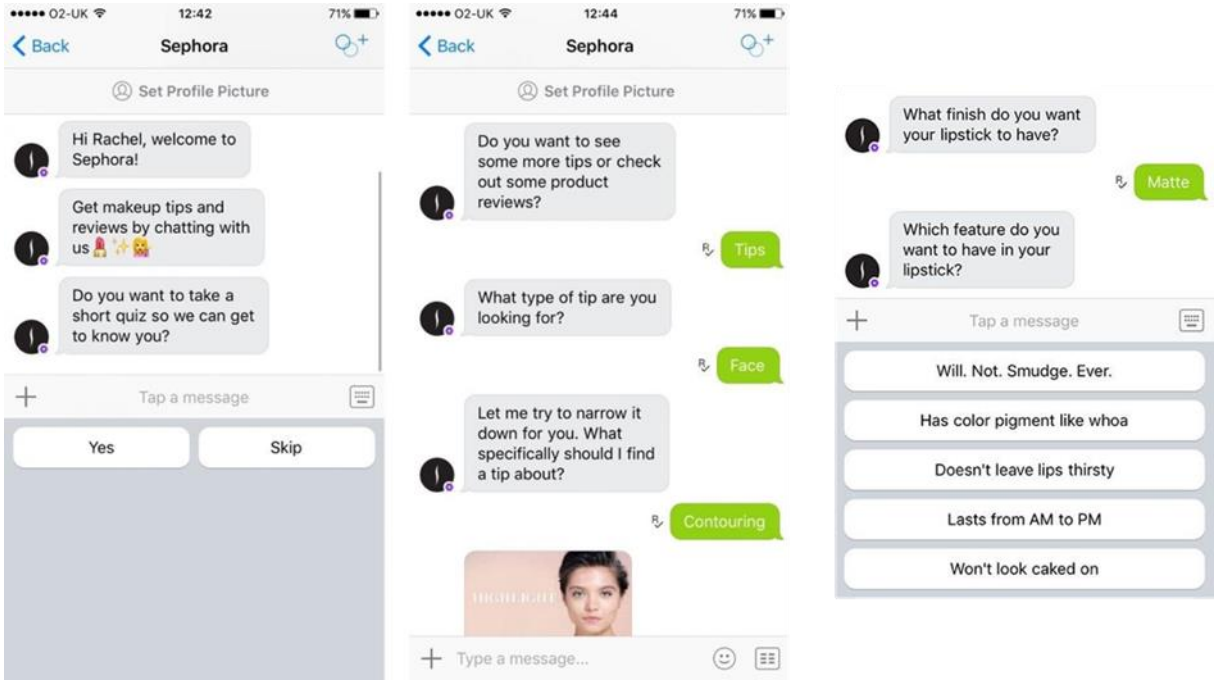

Рисунок 1.3 – Чат-бот Sephora на платформі Kik для персоналізованих рекомендацій

H&M та Burberry представляють ще одну групу найкращих прикладів чат-ботів у цій галузі. Працюючи на платформі обміну повідомленнями, вони надають рекомендації та демонструють вбрання. Ці чат-боти також розміщують посилання на товари на сайті та виконують роль персональних стилістів для клієнтів, які залишаються вдома.

Обидва боти функціонують як цифрові помічники, які перетворюють звичайний онлайн-шопінг на персоналізовану та естетично привабливу взаємодію з брендом. Наприклад, чат-бот H&M запитує про вподобання щодо стилю одягу, кольорової гами та типу подій, для яких підбирається вбрання, після чого формує добірку речей, що відповідають зазначеним критеріям.

Рисунок 1.4 демонструє роботу чат-бота H&M, який надає персоналізовані рекомендації та показує вбрання. Цей бот виконує функцію персонального стиліста, допомагаючи клієнтам здійснювати покупки, не виходячи з дому.

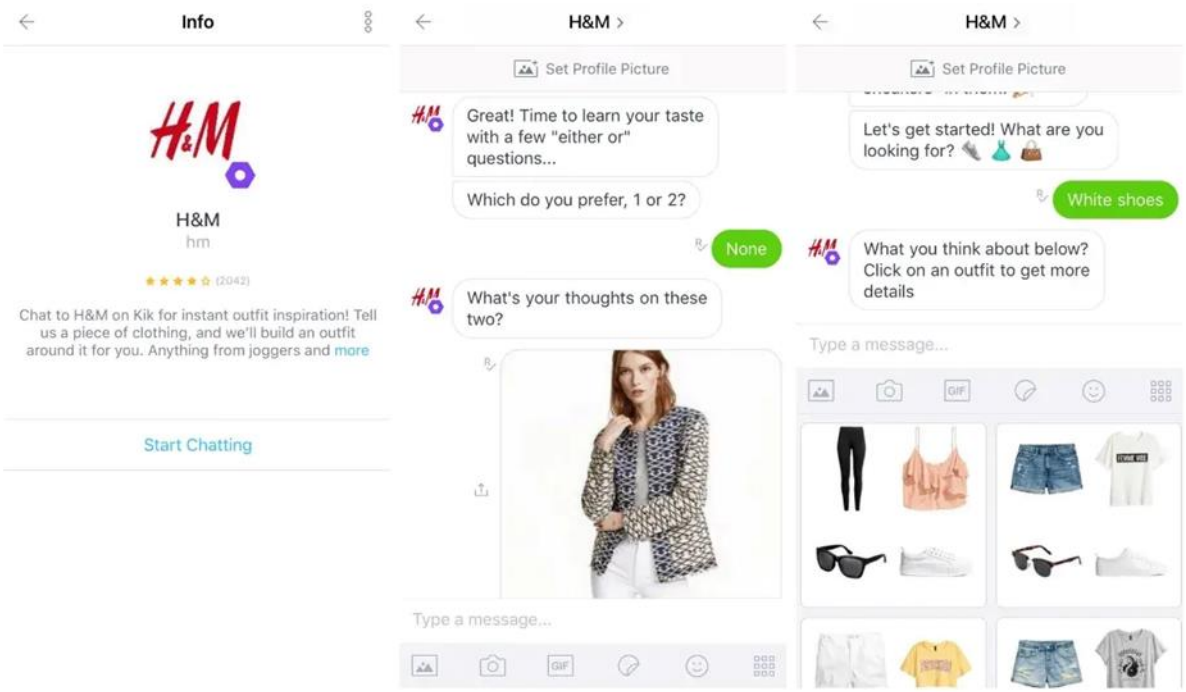

Рисунок 1.4 – Чат-бот H&M як персональний стиліст

Як наслідок, персоналізація обслуговування клієнтів за допомогою ШІ стала ключовим інструментом у побудові унікального користувацького досвіду в електронній комерції. Завдяки точному аналізу поведінки клієнтів, компанії можуть не лише пропонувати релевантні товари, а й формувати глибші емоційні зв’язки з покупцями. Це сприяє підвищенню лояльності, зростанню повторних продажів і створенню конкурентної переваги, що є надзвичайно важливим у динамічному та насиченому онлайн-середовищі.

1.1.3 Аналіз та прогнозування

Штучний інтелект також відіграє ключову роль у прогнозуванні попиту та управлінні запасами на складах. Збір великих обсягів даних про покупців, тенденції на ринку, сезонні коливання, акційні пропозиції та інші фактори дозволяє ШІ створювати точні прогнози щодо майбутнього попиту на певні товари.

Завдяки таким прогнозам компанії можуть краще управляти своїми запасами, скорочуючи витрати на зберігання товарів і знижуючи ризик дефіциту. Автоматизовані системи, що використовують ШІ, можуть навіть самостійно замовляти необхідні товари у постачальників на основі попиту, що сприяє зниженню операційних витрат та поліпшенню логістики.

Управління запасами, засноване на даних, допомагає не тільки оптимізувати бізнес-процеси, але й уникнути перевантаження складів або недостатнього забезпечення популярних товарів, що може призвести до втрати доходу.

Аналіз поведінки споживачів є ще однією важливою функцією, яку забезпечує ШІ в електронній комерції. Завдяки інтеграції з веб-аналітичними інструментами та системами, що базуються на машинному навчанні, компанії можуть отримувати глибоке розуміння того, як користувачі взаємодіють з платформою, які елементи веб-сайту привертають їхню увагу, і чому певні продукти є популярними.

Цей тип аналізу дозволяє розробити точні маркетингові стратегії, покращити клієнтський досвід та оптимізувати ціноутворення. Прогнозування поведінки споживачів також дає змогу створювати більш цільові рекламні кампанії, що максимізує ймовірність конверсії – перетворення потенційного клієнта на реального покупця.

Штучний інтелект також активно використовуються для покращення маркетингових стратегій у сфері електронної комерції. Маркетингові кампанії на основі ШІ дозволяють оптимізувати рекламу, націлюючи її на конкретну аудиторію з урахуванням її інтересів і попередніх покупок. За допомогою таких інструментів, як автоматизовані рекламні кампанії, ретаргетинг, розумні банери та персоналізовані електронні листи, компанії можуть ефективно впливати на покупців.

Таким чином, ШІ дозволяє створювати точні маркетингові кампанії, які можуть змінюватися в режимі реального часу залежно від поведінки користувачів і актуальних трендів.

1.2 Основи обробки природньої мови

1.2.1 Основні завдання обробки природної мови

NLP є однією з основних гілок штучного інтелекту, яка зосереджена на здатності комп'ютерів розуміти, інтерпретувати та генерувати людську мову у формах, зручних для обробки та аналізу. Вона охоплює широкий спектр технологій, алгоритмів і методів, що дозволяють автоматизувати розуміння текстових даних, що є важливим для таких застосувань, як чат-боти, переклад текстів, пошукові системи, система рекомендацій та багато інших.

Одним з головних завдань обробки природної мови є переклад тексту з однієї мови на іншу (машинний переклад), але спектр її завдань значно ширший. До основних задач NLP можна віднести:

– токенізація – процес поділу тексту на складові частини, такі як слова, фрази чи інші одиниці, які є важливими для аналізу, цей базовий етап дозволяє системам розуміти структуру тексту та виділяти значущі елементи для подальшого аналізу;

– часткове позначення мови (POS-tagging) – це визначення частин мови для кожного слова в реченні. Наприклад, для речення «Собака бігає по парку» система NLP визначить, що «Собака» – це іменник, «бігає» – дієслово, а «по парку» – це пре-позиційна фраза;

– аналіз синтаксичної структури – це процес розпізнавання граматичних зв'язків у тексті. Завдяки йому система може зрозуміти, які слова є підметом, присудком, об'єктами тощо, а також визначити, як ці слова взаємодіють між собою в контексті;

– визначення смислового значення – складне завдання, яке передбачає розуміння значення слів і фраз у контексті, для цього застосовуються методи семантичного аналізу та векторизації тексту;

– розпізнавання та аналіз емоцій (Sentiment Analysis) – визначення емоційного забарвлення тексту. Це може бути корисним, наприклад, для аналізу відгуків користувачів чи коментарів у соцмережах, що дозволяє зрозуміти, чи позитивне чи негативне ставлення до продукту чи послуги;

– ідентифікація сутностей (Named Entity Recognition, NER) – визначення важливих сутностей у тексті, таких як імена людей, організацій, місця, дати тощо. Це дозволяє системам ефективно вилучати ключову інформацію з великих масивів текстових даних.

1.2.2 Алгоритми та методи NLP

Для вирішення завдань обробки природної мови використовуються різноманітні методи та алгоритми, що базуються на машинному навчанні та статистичних моделях:

– класичні алгоритми – перші підходи до NLP ґрунтувались на класичних методах обробки даних, таких як регулярні вирази, словники і правила. Наприклад, у системах на основі правил використовуються попередньо визначені лексичні й граматичні правила для виконання певних завдань; вони можуть бути корисними в простих випадках, але мають обмеження, оскільки не враховують контексту;

– статистичні методи – в останні десятиліття все більше застосовуються статистичні підходи, зокрема моделі, які ґрунтуються на ймовірностях. Це включає в себе методи, що застосовують ймовірнісні моделі, такі як наївний байєсівський класифікатор або Hidden Markov Models (HMM). Статистичні моделі дозволяють розпізнавати структури і закономірності в текстах на основі оброблених даних і статистичних зв’язків між елементами мови;

– моделі на основі машинного навчання – останнім часом найбільш популярними стали методи машинного навчання, зокрема навчання з учителем (Supervised Learning). Вони дозволяють системам навчатися на

великій кількості текстових даних, що забезпечує більшу точність у виконанні завдань NLP. Такі моделі можуть використовувати різноманітні алгоритми, зокрема нейронні мережі, для розпізнавання патернів у тексті;

– глибинне навчання та трансформери – останнім досягненням у NLP стало впровадження методів глибинного навчання, зокрема трансформерів (наприклад, BERT (Bidirectional Encoder Representations from Transformers – двонаправлене кодування представлень на основі трансформерів), GPT (Generative Pre-trained Transformer – генеративний трансформер з попереднім навчанням)), які здатні працювати з великими текстовими даними та контекстом. Трансформери дозволяють моделі враховувати як попередній, так і наступний контекст слів у тексті, що значно покращує точність обробки мови. Вони дозволяють створювати моделі, здатні виконувати завдання NLP на високому рівні, включаючи переклад текстів, генерацію текстів та аналіз емоцій.

1.2.3 Застосування NLP в різних сферах

Технології NLP мають безліч практичних застосувань у різних сферах:

– чат-боти та віртуальні помічники – NLP є основою для створення ефективних чат-ботів, що здатні вести діалоги з користувачами, розуміти їхні запити та надавати релевантні відповіді. Віртуальні помічники, такі як «Siri», «Alexa» або «Google Assistant», використовують NLP для обробки голосових запитів та виконання команд;

– аналіз текстових даних та обробка великих обсягів інформації – веб-скрейпінг та автоматизований збір текстових даних з Інтернету часто вимагає застосування NLP для обробки та аналізу текстової інформації. За допомогою NLP можна виділяти важливі дані з новинних статей, відгуків клієнтів, технічних специфікацій тощо;

– системи машинного перекладу – NLP використовується в системах машинного перекладу, таких як «Google Translate», для автоматичного перекладу текстів з однієї мови на іншу. Сучасні методи на основі нейронних мереж дозволяють значно покращити якість перекладу порівняно з традиційними статистичними методами;

– пошукові системи – пошукові системи, такі як Google, використовують NLP для кращого розуміння запитів користувачів та надання більш релевантних результатів пошуку. Це включає в себе синтаксичний та семантичний аналіз запитів, щоб правильно інтерпретувати намір користувача;

– аналіз емоцій в текстах – аналіз емоцій є важливим інструментом для вивчення настроїв споживачів. Такі системи застосовують NLP для виявлення позитивних, негативних або нейтральних емоцій в текстах, що дозволяє компаніям краще зрозуміти реакцію користувачів на продукти або послуги.

У сфері електронної комерції технології NLP відіграють важливу роль у підвищенні якості обслуговування та вдосконаленні користувацького досвіду. Наприклад, за допомогою NLP платформи здатні розпізнавати наміри покупців навіть у разі неструктурованих або некоректно сформульованих запитів, що дозволяє краще адаптувати результати пошуку. Крім того, обробка природної мови широко застосовується для аналізу відгуків клієнтів – системи автоматично визначають тематику, емоційне забарвлення та ключові проблеми, згадані в коментарях, що дає змогу компаніям оперативно реагувати на потреби споживачів. NLP також лежить в основі інтелектуальних систем рекомендацій, які аналізують тексти переглянутих товарів або бажань користувача, щоб запропонувати максимально релевантні позиції. У результаті використання таких технологій сприяє підвищенню рівня персоналізації, задоволеності клієнтів і, як наслідок, зростанню обсягів продажів.

Персоналізований помічник покупців «eBay ShopBot» реалізує можливості обробки природної мови для покращення взаємодії користувачів із платформою. На рисунку 1.5 продемонстровано схему взаємодії користувача з персоналізованим помічником «eBay ShopBot».

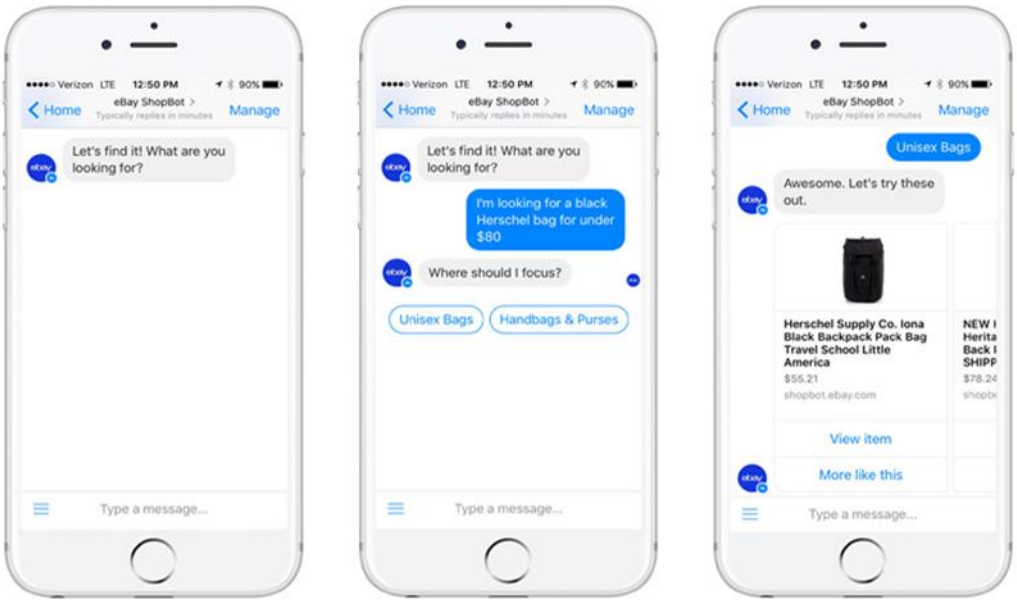

Рисунок 1.5 – Схема взаємодії користувача з персоналізованим помічником «eBay ShopBot»

Бот приймає текстові запити та проводить діалог із користувачем, ставляючи уточнюючі питання для кращого розуміння його намірів. Це дозволяє надати максимально релевантні та персоналізовані рекомендації серед мільярда товарів, представлених на eBay. Такий підхід спрощує процес пошуку та вибору товарів, підвищуючи ефективність і якість обслуговування.

Незважаючи на значні досягнення в області NLP, існує ряд викликів, з якими стикаються дослідники і практики. Одним з основних є проблема розуміння контексту, але перспективи вказують на те, що ця галузь продовжить розвиватися, зокрема завдяки вдосконаленню глибинного навчання та трансформерів.

1.3 Комп’ютерний зір у пошуку товарів: підходи та технології

1.3.1 Еволюція методів комп’ютерного зору

Комп’ютерний зір (Computer Vision) є важливою підгалуззю штучного інтелекту, яка займається аналізом, обробкою та інтерпретацією зображень. У сфері електронної комерції ця технологія активно використовується для розпізнавання об’єктів, пошуку візуально схожих товарів, автоматичного класифікування продукції, контролю якості тощо. Завдяки досягненням у комп’ютерному зорі значно вдосконалився пошук за зображеннями, що особливо актуально для користувачів, які не можуть сформулювати свій запит словами.

На початкових етапах розвитку основними методами були класичні алгоритми, зокрема SIFT (Scale-Invariant Feature Transform – масштабно-інваріантне перетворення ознак) та SURF (Speeded-Up Robust Features – прискорені стійкі ознаки), які базувалися на ручному витяганні ознак. Вони дозволяли знаходити ключові точки на зображеннях, проте мали обмеження – чутливість до змін масштабу, освітлення та ракурсу.

Подальший розвиток привів до впровадження глибокого навчання, зокрема CNN (Convolutional Neural Network – згорткова нейронна мережа). Однією з найуспішніших архітектур стала ResNet (Residual Network – мережа з залишковими зв’язками), яка значно підвищила точність класифікації. CNN дозволили автоматично витягувати ознаки, не потребуючи ручної обробки.

Новий етап – векторні подання зображень. У цьому підході зображення перетворюється на вектор, що дозволяє ефективно знаходити схожі товари. Вагому роль у цьому зіграла модель CLIP (Contrastive Language–Image Pretraining – контрастивне попереднє навчання для роботи з мовою і зображеннями), яка поєднує зображення з текстом в єдиному

семантичному просторі, що дало змогу реалізовувати пошук навіть без текстового опису.

1.3.2 Актуальні технології та практичне застосування

У сучасних системах електронної комерції активно використовуються моделі, що поєднують ефективність і швидкодію. Наприклад, EfficientNet і MobileNet забезпечують хорошу якість класифікації за низьких ресурсів, тому підходять для мобільних додатків. ViT (Vision Transformer – трансформер для зору) пропонують інноваційний спосіб обробки зображень, що базується на аналізі послідовностей патчів, як у тексті.

Інтеграція комп’ютерного зору у вебплатформи дозволяє користувачам завантажити фото товару й отримати візуально схожі пропозиції. Векторні подання товарів заздалегідь генеруються за допомогою згаданих моделей, що дає змогу швидко й точно знаходити збіги.

Вибір моделі залежить від масштабу проєкту: для великих маркетплейсів доцільно застосовувати ResNet або CLIP, тоді як для нішевих проєктів – легкі CNN або MobileNet. У центрі уваги – не лише якість розпізнавання, а й масштабованість рішень.

У майбутньому очікується синергія комп’ютерного зору та мовних моделей, що відкриває нові можливості для персоналізованих рекомендацій, пошуку за змістом та створення «розумних» платформ продажу.

Хоча «eBay ShopBot» володіє багатьма базовими функціями сучасних чат-ботів, такими як підтримка природної мови та розпізнавання голосу, він має низку унікальних характеристик, що вирізняють його серед аналогів. По-перше, користувачам надається можливість створення персоналізованого профілю та налаштувань, що дозволяє боту одразу пропонувати релевантні товари, адаптовані до індивідуальних вподобань.

По-друге, інтегроване розпізнавання зображень дає змогу користувачам фотографувати предмети, наприклад, сорочки, і бот аналізує зображення для пошуку аналогічних товарів на платформі eBay. Таким чином, «eBay ShopBot» значно підвищує зручність і точність підбору продукції. Інтерактивний інтерфейс бота представлено на рисунку 1.6.

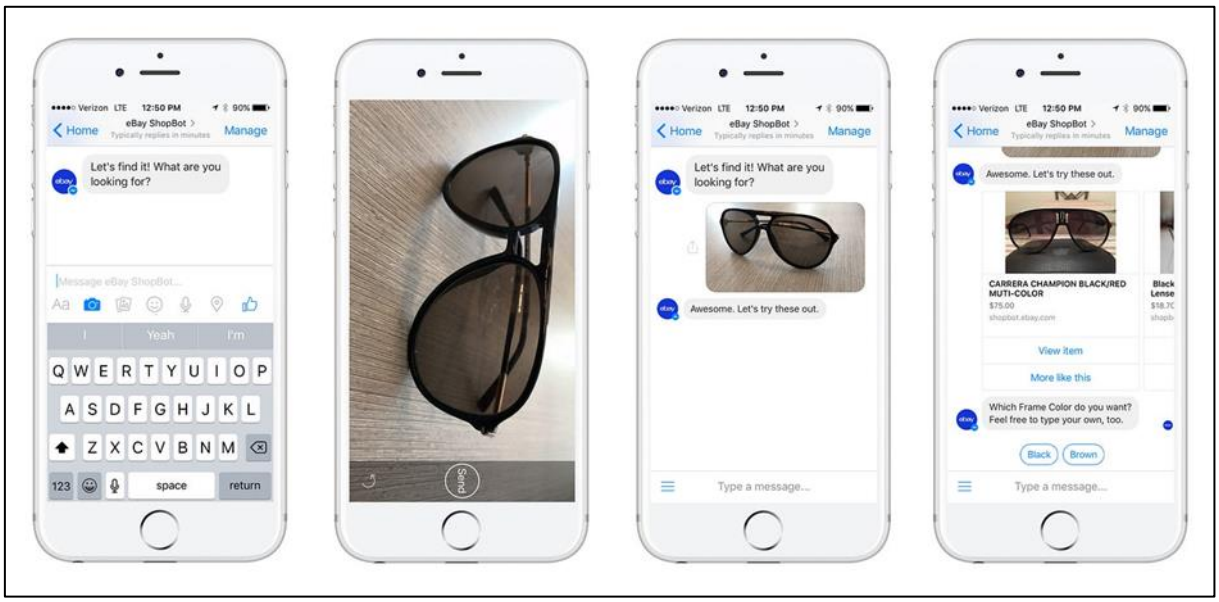

Рисунок 1.6 – Інтерфейс персоналізованого помічника «eBay ShopBot» із функцією розпізнавання зображень

Сучасні технології комп’ютерного зору у поєднанні з методами обробки природної мови створюють нові можливості для покращення користувацького досвіду в електронній комерції. Використання інноваційних моделей та інтеграція багатомодальних рішень дозволяє створювати персоналізовані, швидкі й ефективні інтерфейси, які значно спрощують процес пошуку та вибору товарів. Подальший розвиток цих технологій сприятиме підвищенню якості сервісу та відкриє нові горизонти для автоматизації та інтелектуалізації онлайн-продажів.

1.4 Огляд технологій для створення інтелектуальних чат-ботів

1.4.1 Від шаблонних систем до глибинного навчання

Інтелектуальні чат-боти є одним із ключових напрямів розвитку штучного інтелекту в галузі цифрових сервісів, зокрема електронної комерції. Їхнє завдання – моделювати природну мовну взаємодію з користувачами для автоматизації обслуговування, надання консультацій, навігації каталогом товарів і оформлення замовлень.

Перші чат-боти працювали за допомогою шаблонних правил: прості if/else конструкції, де відповіді прив’язувалися до заданих фраз. Такі рішення не справлялися зі складними чи непередбачуваними запитами й не враховували мовні варіації, тому швидко втратили актуальність.

Значного прориву вдалося досягти завдяки глибокому навчанню та методам векторизації тексту, зокрема word2vec, GloVe (Global Vectors – глобальні вектори) та FastText. Ці технології дозволили представляти слова у вигляді векторів, що зробило можливою побудову більш гнучких і контекстно-залежних моделей обробки мови.

1.4.2 Трансформери та генеративні моделі у чат-ботах

Наступним етапом став розвиток моделей на основі трансформерів – архітектури, яка змінила підхід до обробки мови. Моделі на кшталт BERT, RoBERTa (Robustly optimized BERT approach – оптимізований підхід до BERT), DistilBERT (Distilled BERT – дистильований BERT), T5 (Text-To-Text Transfer Transformer – трансформер для перетворення тексту в текст), а також GPT забезпечили здатність аналізувати весь контекст речення одночасно, що суттєво підвищило якість інтерпретації намірів користувача.

Завдяки цим моделям сучасні чат-боти можуть не лише класифікувати запити, а й:

– генерувати зв’язні й контекстуально релевантні відповіді;

– вести тривалий діалог із пам’яттю на попередні репліки;

– адаптувати стиль мовлення під користувача.

В електронній комерції такі боти активно застосовуються для персоналізації, виявлення інтенцій (намірів), аналізу запитів та генерації рекомендацій. Крім того, поєднання генеративного підходу з логікою класифікації інтенцій дає змогу створювати гібридні системи, здатні поєднувати діалог із діями, як-от перевірка статусу замовлення чи виклик API (Application Programming Interface – інтерфейс прикладного програмування).

1.4.3 Інфраструктура, інструменти та практичне впровадження

З технічного боку реалізація чат-ботів ґрунтується на вебсервісах із використанням таких фреймворків, як Flask або FastAPI на Python. Це дозволяє інтегрувати моделі ШІ у веб або мобільні застосунки.

Для зберігання даних (історії діалогів, товарних векторів тощо) часто застосовують нереляційні бази даних, як-от MongoDB, що дає змогу зберігати гнучкі документи у форматі JSON (JavaScript Object Notation – нотація об’єктів JavaScript).

Крім локального розгортання моделей, великого значення набули хмарні платформи доступу до AI (Artificial Intelligence – штучний інтелект) через API, зокрема OpenAI (ChatGPT), Google Cloud AI та Cohere(GPT-подібна модель з API-доступом).

Ці сервіси дають змогу малому та середньому бізнесу використовувати потужні LLM-моделі (Large Language Model – велика мовна модель) без потреби у власних обчислювальних потужностях.

Окремим компонентом є інтент-класифікатори, реалізовані на основі PyTorch або TensorFlow. Вони відповідають за розпізнавання цілей користувача – від пошуку товару до отримання допомоги.

2 ПОСТАНОВКА ЗАДАЧІ ТА ОБҐРУНТУВАННЯ ТЕХНОЛОГІЙ

2.1 Визначення вимог до функціоналу

У контексті стрімкого розвитку електронної комерції та інтелектуальних систем взаємодії з користувачами, традиційні інтернет-магазини вже не завжди відповідають зростаючим очікуванням споживачів щодо зручності, персоналізації та швидкості обслуговування. Сучасний покупець прагне отримати не лише доступ до широкого асортименту товарів, а й оперативну підтримку, інтуїтивно зрозумілий інтерфейс та інтелектуального помічника, здатного ефективно орієнтувати його у великому масиві інформації. У зв’язку з цим, розробка системи електронної комерції, яка включає інтегрований чат-бот з елементами машинного навчання, є актуальним і доцільним завданням.

У межах проєкту реалізовано інтернет-магазин жіночого одягу, який продає товар зі знижками по країніх ЄС із вбудованим інтелектуальним чат-ботом, функціональність якого охоплює два основні напрями. Перший напрям передбачає обробку текстових запитів користувачів у режимі реального часу. Бот використовує попередньо навчений класифікатор, побудований на основі бібліотеки PyTorch, що дозволяє здійснювати розпізнавання інтенцій, формувати релевантні відповіді та надавати рекомендації щодо товарів, способів оплати, доставки й інших аспектів взаємодії з магазином. У випадках, коли рівень впевненості класифікатора є недостатнім, система автоматично звертається до зовнішнього API від платформи Cohere, яка забезпечує генерацію більш змістовної відповіді з використанням попередньо завантажених структурованих даних про магазин. Такий комбінований підхід дозволяє поєднати ефективність локальної моделі з адаптивністю генеративного ШІ, що значно підвищує якість обслуговування користувачів.

Друга ключова функція чат-бота полягає у виконанні пошуку товарів за зображенням. У даному випадку реалізовано можливість обробки візуального запиту користувача шляхом завантаження зображення, наприклад, елемента одягу. Завдяки використанню моделі CLIP, яка поєднує можливості комп’ютерного зору та обробки текстової інформації, система здатна зіставляти векторні представлення зображення з векторними описами товарів у базі даних. У результаті користувачу пропонується список товарів, які візуально схожі на об'єкт на завантаженому зображенні. Це значно спрощує пошук для тих користувачів, які не володіють точною назвою товару або не мають змоги його описати в деталях.

Веб-інтерфейс розробляється із застосуванням бібліотеки React.js, що дозволяє створити гнучку та інтерактивну клієнтську частину додатку з високою швидкістю відгуку. Для зберігання даних використовується база даних MongoDB, яка завдяки своїй документно-орієнтованій структурі надає змогу зручно управляти інформацією про користувачів, товари, історію замовлень та запити до чат-бота.

Програмний продукт має забезпечити повноцінну взаємодію користувача з інтерфейсом магазину, підтримку персоналізованого сервісу за допомогою текстового й візуального запиту, а також реалізацію функціоналу чат-бота як ключового елементу покращення користувацького досвіду. Усі зазначені компоненти повинні працювати узгоджено, забезпечуючи як масштабованість системи, так і стабільність її функціонування у разі зростання навантаження.

2.2 Архітектура системи: взаємодія компонентів

2.2.1 Загальна архітектура

Архітектура системи інтернет-магазину з інтегрованим чат-ботом складається з кількох ключових компонентів, що взаємодіють між собою

для забезпечення ефективної роботи платформи. Основними елементами архітектури є фронтенд (клієнтська частина), чат-бот і база даних, кожен з яких має свою унікальну роль у функціонуванні системи.

Фронтенд реалізований за допомогою технології React.js. Це клієнтська частина інтерфейсу, через яку користувачі взаємодіють з інтернет-магазином. Веб-інтерфейс дозволяє відображати товари, кошик, список вподобаних товарів, а також забезпечує інтерактивне спілкування з чат-ботом. Завдяки React.js користувач отримує зручний, швидкий і адаптивний інтерфейс, що автоматично оновлюється у відповідь на зміни стану системи, створюючи комфортні умови для покупки товарів.

Чат-бот, розроблений з використанням Python, PyTorch та CLIP, виконує дві основні функції: обробляє текстові запити користувачів і шукає товари за зображеннями. Технології машинного навчання, зокрема використання попередньо навченої моделі для класифікації інтенцій користувачів та моделі CLIP для пошуку товарів за зображеннями, забезпечують точні і швидкі відповіді на запити користувачів, покращуючи їхній досвід взаємодії з магазином.

Для зберігання даних використовується MongoDB. Це документно-орієнтована база даних, яка забезпечує зберігання та організацію даних про товари, користувачів, їхні замовлення та запити до чат-бота. Оскільки інтернет-магазин може містити великі обсяги даних, MongoDB забезпечує гнучкість і масштабованість системи, дозволяючи ефективно управляти інформацією навіть при зростанні навантаження.

Кожен з цих компонентів працює в тісній взаємодії, щоб забезпечити стабільну роботу інтернет-магазину, надаючи користувачам можливість швидко знаходити потрібні товари, отримувати підтримку через чат-бота та здійснювати покупки з максимальною зручністю. Також ця взаємодія дає змогу оперативно обробляти великі обсяги інформації в режимі реального часу, забезпечуючи індивідуалізований підхід до кожного користувача.

2.2.2 Взаємодія компонентів

Взаємодія компонентів системи є ключовою для забезпечення безперебійної роботи інтернет-магазину та інтегрованого чат-бота. Кожен компонент має свою специфічну роль, але вони всі працюють злагоджено для надання користувачеві найкращого досвіду взаємодії з платформою.

Клієнтська сторона взаємодіє з чат-ботом через інтерфейс, де користувач може здійснювати різні дії, такі як пошук товарів, перегляд детальних описів, додавання товарів до кошика або взаємодія з чат-ботом для отримання підтримки. Коли користувач вводить текстовий запит або завантажує зображення для пошуку товару, фронтенд передає ці дані на сервер. Запити текстового типу обробляються чат-ботом, який здійснює необхідний пошук в базі даних або через зовнішні сервіси. У випадку пошуку товарів за зображенням, фото товару, яке завантажує користувач, передається на сервер для подальшої обробки за допомогою моделі CLIP.

Чат-бот, в свою чергу, взаємодіє з базою даних, де зберігаються всі необхідні дані про товари, користувачів та їхні запити. Коли бот отримує текстовий запит, він звертається до бази даних для отримання інформації, необхідної для формування відповіді або пропозиції товарів. Це може бути інформація про наявність товару, методи доставки, умови оплати тощо. У випадку пошуку за зображенням, чат-бот використовує модель CLIP, щоб порівняти векторні представлення завантаженого зображення з даними, що зберігаються у базі, і знайти схожі товари.

Інтерфейс користувача також безпосередньо взаємодіє з базою даних через сервер API для отримання даних про товари, вподобані товари та історії покупок. Вся інформація про користувачів, їхні уподобання та кошики покупок зберігається в базі даних, що дозволяє персоналізувати досвід покупок для кожного користувача. Ця інтеграція дозволяє синхронізувати дії користувача на фронтенді з даними, що зберігаються в базі.

Коли чат-бот не може знайти достатньо інформації для відповіді з використанням внутрішніх моделей, він звертається до зовнішнього API від платформи Cohere. Це забезпечує генерацію більш змістовних і точних відповідей, оскільки API використовує попередньо завантажені структуровані дані магазину. Така інтеграція з зовнішніми сервісами дозволяє чат-боту бути більш гнучким і ефективним у вирішенні різноманітних запитів користувачів.

Ця комплексна система взаємодії між фронтендом, чат-ботом, базою даних і зовнішніми сервісами дозволяє забезпечити високий рівень автоматизації та персоналізації інтернет-магазину, що значно покращує користувацький досвід.

2.3 Вибір технологій для реалізації проєкту

У процесі розробки інтернет-магазину з інтегрованим інтелектуальним чат-ботом для забезпечення високої ефективності, зручності розробки та масштабованості системи було вибрано певний стек технологій. Кожна з обраних технологій має своє обґрунтування, яке відповідає вимогам проекту, зокрема забезпечує оптимальну взаємодію між компонентами, зручність розробки та надійність.

React.js був обраний для розробки клієнтської частини інтерфейсу завдяки своїй високій продуктивності та швидкості відгуку. Це дозволяє створювати динамічні та інтерактивні веб-додатки, оптимізуючи процес оновлення інтерфейсу за допомогою віртуального DOM (Document Object Model – модель об'єкта документа). Такий підхід мінімізує кількість змін, що вимагають переробки всього дерева елементів. Крім того, React.js використовує модульну структуру, що дозволяє розділити інтерфейс на незалежні компоненти, полегшуючи розробку та підтримку. Це також дозволяє легко інтегрувати сторонні бібліотеки та сервіси, такі як Redux для керування станом або React Router для навігації. Завдяки гнучкості React.js,

розробка адаптивних інтерфейсів для мобільних пристроїв стає значно легшою, що є важливим для інтернет-магазинів.

MongoDB була обрана через свою здатність ефективно працювати з великими обсягами даних і гнучкість у зберіганні інформації. Використання документно-орієнтованої бази даних дозволяє зберігати різноманітні типи даних у вигляді JSON-подібних документів, що є ідеальним для збереження інформації про товари, користувачів та історії покупок. Завдяки високій масштабованості, MongoDB може зростати разом з бізнесом і справлятися з великими навантаженнями, що є важливим для інтернет-магазину з постійно зростаючим обсягом даних. Вбудовані механізми реплікації і поширення забезпечують високу доступність даних, що необхідно для безперебійної роботи інтернет-магазину.

Python був вибраний як основна мова програмування для розробки чат-бота через свою простоту, гнучкість та наявність великого набору бібліотек для роботи з машинним навчанням і обробкою природної мови. Мова активно використовується в галузі штучного інтелекту та машинного навчання, що дозволяє швидко розробляти і адаптувати моделі для розпізнавання інтенцій та аналізу тексту. Python також підтримує інтеграцію з іншими технологіями, що дає можливість реалізувати складні функціональні можливості для чат-бота, зокрема для пошуку товарів за зображенням. Завдяки простому та читабельному коду, Python є зручним для швидкої розробки прототипів і подальшої оптимізації продукту.

PyTorch була вибрана як основний інструмент для реалізації моделей машинного навчання через свою гнучкість і зручність у використанні. Ця бібліотека підтримує динамічні графи обчислень, що спрощує налаштування та оптимізацію моделей для конкретних завдань. PyTorch має простий синтаксис і широкий набір інструментів, що дозволяє ефективно працювати з глибокими нейронними мережами, необхідними для обробки тексту та зображень. Вибір PyTorch обумовлений також її здатністю інтегруватися з іншими бібліотеками та технологіями, що дозволяє

створювати комплексні моделі для розпізнавання інтенцій користувачів і пошуку товарів.

CLIP була обрана для реалізації пошуку товарів за зображенням завдяки її унікальній здатності поєднувати текстову та візуальну інформацію. CLIP створює векторні представлення як для текстових описів, так і для зображень, що дозволяє здійснювати порівняння зображень товарів з їх текстовими описами в базі даних. Цей підхід значно спрощує пошук товарів для користувачів, які можуть не точно описувати товар текстом, але мають його візуальне зображення. CLIP також підтримує попередньо навчену модель, що дозволяє швидко інтегрувати її в систему та знижує витрати часу на розробку власних моделей.

Загалом, вибір цих технологій забезпечує ефективну реалізацію та інтеграцію компонентів інтернет-магазину з чат-ботом, дозволяючи досягти високого рівня функціональності, ефективності та масштабованості системи.

2.4 Алгоритм реалізації інтелектуального чат-бота: аналіз інтенцій і пошук зображень

2.4.1 Обробка текстових запитів

Першим етапом реалізації чат-бота є обробка текстових запитів користувачів. Чат-бот отримує повідомлення у вигляді тексту від користувача, який може запитувати про різні аспекти роботи інтернет-магазину, такі як наявність товару, варіанти доставки, оплату тощо. Для того щоб надати релевантні відповіді, чат-бот використовує систему розпізнавання інтенцій, що включає кілька ключових етапів.

На першому етапі запит користувача передається до моделі машинного навчання, побудованої на бібліотеці PyTorch. Це може бути модель класифікації тексту, яка визначає мету повідомлення

користувача (наприклад, запит на інформацію про наявність товару, доставку чи умови оплати). Модель класифікації інтенцій використовує попередньо навчений набір даних для розпізнавання типів запитів, що дозволяє чат-боту класифікувати запит і обрати відповідну стратегію для надання відповіді.

Якщо намір чітко розпізнаний, чат-бот одразу може надати відповідь, звернувшись до бази даних для отримання актуальної інформації про товар, доступність, варіанти оплати чи доставки. У випадку, якщо класифікатор не може впевнено визначити інтенцію через низьку впевненість, система здійснює додаткове звернення до зовнішнього API, наприклад, платформи Cohere, для генерації змістовніших відповідей.

Після визначення намірів чат-бот здійснює генерацію текстової відповіді. Якщо інтенція стосується товару або специфікації продукту, бот звертається до бази даних MongoDB для пошуку відповідної інформації, як-от назви товару, ціни, доступності на складі або навіть надає пропозиції схожих товарів. У разі необхідності, додатково можуть бути надані посилання на сторінки товару чи інструкції для здійснення покупки.

Після формування відповіді вона передається користувачу у вигляді текстового повідомлення через інтерфейс чат-бота. Завдяки такій архітектурі чат-бот здатний швидко і точно реагувати на запити, покращуючи якість обслуговування клієнтів та підвищуючи рівень їх задоволеності.

2.4.2 Пошук товарів за зображенням

Другим основним етапом роботи чат-бота є пошук товарів за зображеннями. Ця функція надає можливість користувачам завантажувати фотографії товарів, які вони хочуть знайти в магазині, наприклад, елементів одягу. Алгоритм пошуку за зображенням реалізується через модель CLIP,

що поєднує обробку зображень та тексту для здійснення порівняння і пошуку схожих товарів.

Першим кроком є завантаження користувачем зображення товару. Зображення передається через фронтенд на сервер, де воно попередньо обробляється для підготовки до аналізу. Це може включати зміни розміру або нормалізацію, щоб зображення відповідало вимогам моделі CLIP.

Модель CLIP поєднує текстову та візуальну інформацію для створення векторних представлень як для зображень, так і для текстових описів. Завантажене зображення конвертується в векторне представлення, яке порівнюється з векторними представленнями товарів, що вже є в базі даних.

CLIP здійснює цей процес шляхом виведення векторів для кожного товару з бази даних, що включає в себе візуальні характеристики та текстовий опис. Зображення користувача порівнюється з цими векторами, і система вибирає найбільш релевантні товари, які за своєю візуальною характеристикою схожі на надане зображення.

Після порівняння векторних представлень система видає список товарів, які найбільше відповідають запиту користувача. Результати пошуку містять товари, що мають схожі візуальні характеристики. Користувачеві надається можливість переглядати деталі кожного товару, а також додавати його до кошика чи позначати як вподобане.

Для забезпечення ефективної роботи чат-бота з іншими компонентами інтернет-магазину важливою є інтеграція з базою даних. Чат-бот, залежно від запиту, взаємодіє з MongoDB для отримання необхідної інформації про товари, збереження запитів користувачів, а також для зберігання векторних представлень товарів у разі пошуку за зображенням.

При необхідності, якщо чат-бот не може знайти відповіді в межах своєї локальної моделі, система звертається до зовнішнього API для отримання додаткової інформації або генерації змістовніших відповідей, що покращує якість обслуговування користувачів.

Таким чином, алгоритм реалізації інтелектуального чат-бота об'єднує кілька складних технологій для досягнення високої ефективності в обробці текстових запитів і пошуку товарів за зображенням. Це дозволяє значно підвищити рівень взаємодії з користувачами і покращити їхній досвід покупки в інтернет-магазині.

2.5 Оцінка альтернативних рішень та обґрунтування вибору підходу

У процесі проєктування інтернет-магазину з інтегрованим інтелектуальним чат-ботом було розглянуто низку альтернативних підходів до реалізації ключових функціональних складових системи: обробки текстових запитів, пошуку за зображеннями, зберігання даних та створення веб-інтерфейсу. Вибір остаточного технологічного стеку ґрунтувався на аналізі ефективності, гнучкості, доступності інструментів, а також на здатності забезпечити масштабованість і підтримку інтелектуальних функцій системи. Нижче розглянуто основні альтернативи та обґрунтовано переваги обраного рішення.

2.5.1 Альтернативи в реалізації чат-бота

Одним з можливих варіантів створення чат-бота було використання готових платформ, таких як Dialogflow від Google або Microsoft Bot Framework. Ці інструменти надають зручний інтерфейс для створення сценаріїв, вбудовані механізми обробки природної мови та інтеграції з месенджерами. Проте ці рішення часто мають обмеження у гнучкості, складнощі з індивідуальним налаштуванням моделей, а також менш контрольований доступ до внутрішніх механізмів обробки запитів. Крім того, у випадку складніших функцій, таких як пошук за зображенням, ці платформи не надають необхідного рівня інтеграції з комп’ютерним зором.

На відміну від цього, власна реалізація чат-бота на Python із застосуванням PyTorch дала змогу гнучко керувати всіма етапами аналізу запиту, налаштовувати класифікатор інтенцій на основі конкретних потреб магазину, а також реалізувати адаптивний механізм вибору відповіді з можливістю звернення до зовнішніх API. Такий підхід дозволяє поєднати точність власної моделі з адаптивністю генеративного ШІ, що є важливою перевагою.

Для реалізації функції пошуку товарів за зображенням було розглянуто кілька альтернативних підходів і технологій у сфері комп'ютерного зору. Серед них – використання популярних моделей глибокого навчання, таких як ResNet і Inception, які добре зарекомендували себе для витягування ознак (feature extraction) із зображень. Ці моделі можуть бути адаптовані для створення векторних представлень товарів, що дозволяє порівнювати їх за візуальною схожістю. Крім того, розглядалися варіанти побудови системи пошуку з нуля із застосуванням бібліотек OpenCV, що надають широкі можливості для обробки та аналізу зображень, або інтеграції готових хмарних сервісів, таких як Amazon Rekognition чи Google Vision API, які забезпечують швидку і точну ідентифікацію об'єктів і класифікацію.

Проте ці підходи мають обмеження, особливо коли йдеться про мультимодальний пошук – порівняння зображень із текстовими описами товарів. У таких випадках важливо мати модель, що може ефективно «розуміти» і співставляти дані різних типів. Саме тому вибір було зроблено на користь моделі CLIP від OpenAI. Ця модель створює єдиний векторний простір для обробки як зображень, так і текстів, що дозволяє виконувати пошук, навіть якщо користувач не може точно сформулювати запит словами або взагалі не надає текстового опису. Це особливо корисно для інтернет-магазину одягу, де важливішою є візуальна відповідність товарів – наприклад, користувач може шукати за фотографією схожого виробу без необхідності детально описувати його характеристики.

Завдяки використанню CLIP реалізується більш гнучкий і точний пошук, який підвищує якість взаємодії користувачів із платформою та сприяє збільшенню конверсії за рахунок швидкого знаходження потрібних товарів за візуальними ознаками. Крім того, інтеграція мультимодального пошуку дозволяє створити більш інтуїтивний інтерфейс, що підвищує загальний рівень задоволеності клієнтів і покращує конкурентоспроможність інтернет-магазину.

2.5.2 Аналіз варіантів зберігання даних і вибору фреймворку для клієнтської частини

Серед альтернатив до React.js розглядалися Vue.js та Angular. Vue.js пропонує простіший синтаксис і легший вхід для початківців, а Angular – потужний набір інструментів для масштабних проєктів. Проте React.js став найкращим вибором завдяки своїй популярності, активній спільноті, високій продуктивності, компонентному підходу та широкій підтримці інструментів інтеграції. Його структура дозволяє легко створювати динамічні сторінки, адаптивний інтерфейс і забезпечити плавну взаємодію з бекендом (серверна частина) та чат-ботом.

Як базу даних для системи розглядалися також реляційні системи керування базами даних (СКБД), такі як PostgreSQL (Structured Query Language – мова структурованих запитів) або MySQL, які відзначаються високою надійністю та структурованістю. Основні характеристики та переваги розглянутих СКБД узагальнено у таблиці 2.1. Проте для задач, пов’язаних із зберіганням гнучких та різнорідних об'єктів, як-от інформація про товари, запити, користувачів, а також векторні представлення для пошуку за зображенням, документно-орієнтована база MongoDB виявилася значно зручнішою.

MongoDB забезпечує високу гнучкість схеми, що дозволяє змінювати структуру збережених документів без складних міграцій. Крім того, вона

добре масштабується і підтримує шардінг, що є важливою перевагою у випадку зростання навантаження на систему.

Таблиця 2.1 – Порівняння баз даних

База даних	Тип	Переваги	Недоліки
MongoDB	NoSQL (документна)	Гнучкість схеми, масштабованість, JSON-подібні документи.	Слабка підтримка складних транзакцій.
PostgreSQL	SQL (реляційна)	Потужні запити, підтримка транзакцій, розширення.	Складніша настройка масштабування.
MySQL	SQL (реляційна)	Широке розповсюдження, надійність, ефективність.	Може бути повільнішим у NoSQL-сценаріях.
Firebase	NoSQL (реального часу)	Синхронізація в реальному часі, готові бекенд-функції.	Обмежена гнучкість, залежність від Google.

Узагальнюючи, вибір технологій для розробки системи залежить від специфічних вимог проекту. React.js виявився оптимальним вибором завдяки високій продуктивності та здатності легко інтегруватися з іншими інструментами. Щодо баз даних, MongoDB виявилася значно більш гнучкою для роботи з різнорідними даними, забезпечуючи масштабованість і простоту змінюваності структури.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

3.1 Структура та моделювання бази даних MongoDB

Одним із ключових компонентів розробки веб-застосунку «Інтернет-магазин з чат-ботом на основі машинного навчання» є база даних, схема якої наведена на рисунку 3.1.

Рисунок 3.1 – Схема бази даних

Вона виконує роль централізованого сховища, у якому зберігаються всі дані, необхідні для функціонування системи: інформація про користувачів, товари, кошики, замовлення, рекламні слайди та токени автентифікації. Окрім того, база даних забезпечує взаємодію з чат-ботом, який реалізовано за допомогою мови програмування Python і підключено до тієї ж бази даних через інтерфейс pymongo (офіційна бібліотека для роботи з базою даних MongoDB у Python.). Це дозволяє боту оперативно отримувати інформацію з магазину, наприклад, шукати товари або підказувати користувачу, як оформити замовлення.

Для реалізації бази даних було обрано документоорієнтовану систему керування базами даних MongoDB, яка чудово підходить для веб-застосунків з високою динамікою структури даних та потребою в масштабуванні. В основі структури бази лежать шість основних колекцій: Users, Cards, Cart, Orders, MainSlides та Token, кожна з яких виконує свою окрему функцію у загальній архітектурі системи.

Колекція користувачів містить всю необхідну інформацію для авторизації, персоналізації та керування правами доступу. Кожен користувач може додавати товари до улюбленого, зберігати адресу доставки та мати свій власний кошик. Товари, які формують асортимент магазину, представлені в окремій колекції, де кожна картка товару зберігає детальний опис, зображення, бренд, доступні розміри, тканину, ціну, рейтинг, а також кількість разів, коли товар був доданий до замовлень. Це дозволяє формувати рекомендації та аналітику на основі популярності товарів.

Оформлені користувачами замовлення зберігаються в колекції Orders, яка містить дані про дату замовлення, адресу доставки, список товарів, загальну вартість та статус виконання доставки. Всі товари в замовленні є посиланнями на відповідні об’єкти з колекції товарів, що дозволяє уникати дублювання інформації та забезпечує узгодженість даних.

Окрему роль у структурі займає колекція Cart, яка реалізує збереження тимчасових списків покупок. Кожен кошик належить певному

користувачу та може містити довільну кількість товарів із вказаною кількістю одиниць. Таким чином, структура кошика дає змогу динамічно оновлювати вміст у режимі реального часу при взаємодії користувача з інтерфейсом магазину.

Для збереження банерів, які відображаються на головній сторінці сайту, передбачено колекцію MainSlides, де зберігаються зображення рекламного характеру. Ця інформація не пов’язана безпосередньо з іншими сутностями, але використовується для візуального оформлення інтерфейсу.

З метою реалізації захищеної авторизації система також зберігає токени оновлення у спеціальній колекції Token. Вона дозволяє користувачу залишатися авторизованим протягом тривалого часу, не порушуючи принципів безпеки.

Всі ці колекції взаємодіють між собою через унікальні ідентифікатори, формуючи логічну структуру, яка дозволяє ефективно організовувати, читати й оновлювати дані в реальному часі. Особливістю даної бази є її сумісність з чат-ботом, який через API отримує доступ до даних, аналізує запити користувачів та формує відповідь на основі машинного навчання. Це дає змогу реалізувати інтелектуального помічника для користувачів магазину, що значно покращує взаємодію з системою.

Усі колекції реалізовані за допомогою бібліотеки Mongoose. Для ілюстрації структури надано схему колекції Cards, яка є найбільш репрезентативною. Приклад Mongoose-схеми подано в додатоку А. Інші схеми, такі як Users, Orders, Cart, реалізовані за аналогічним принципом, з урахуванням особливостей відповідних даних.

Розроблена схема бази даних повністю відповідає функціональним вимогам проєкту, підтримує гнучку структуру даних та легко масштабується в умовах зростання навантаження. MongoDB як технологія дозволяє забезпечити стабільну роботу застосунку, високу продуктивність і простоту в подальшій розробці.

3.2 Реалізація клієнтської частини інтернет-магазину (React.js)

Клієнтська частина інтернет-магазину реалізована за допомогою бібліотеки React.js, що забезпечує компонентно-орієнтований підхід до побудови інтерфейсу користувача. Основною метою цієї частини системи є забезпечення інтуїтивно зрозумілої та швидкої взаємодії користувача з функціоналом магазину – перегляд товарів, додавання до кошика, авторизація, спілкування з чат-ботом тощо.

Проєкт ініціалізовано з використанням Vite – сучасного інструменту для швидкої розробки веб-додатків, що забезпечує значно кращу продуктивність порівняно з Create React App. Візуальна частина реалізована без використання CSS-фреймворків (Cascading Style Sheets – каскадні таблиці стилів) типу Tailwind; замість цього використано модульний CSS (CSS Modules), що забезпечує ізольованість стилів для кожного компонента.

Основні компоненти клієнтської частини включають головну сторінку, на якій розміщуються слайдери з банерами (отримуються з API на основі колекції MainSlides у базі даних), список популярних товарів та навігаційне меню. Макет головної посадкової сторінки (landing page) зображений в додатку Б. Сторінка каталогу товарів реалізована з фільтрацією, сортуванням та пагінацією, а запити до товарів надсилаються до серверного API через асинхронні HTTP-запити (HyperText Transfer Protocol – протокол передавання гіпертексту). Окрема сторінка товару відображає розширену інформацію про товар, його фото, доступні розміри, опис, рейтинг та інші характеристики. На цій сторінці також є функція додавання товару до кошика або улюбленого списку.

Також є кошик та сторінка оформлення замовлення, де користувач може переглядати додані товари, змінювати кількість, видаляти позиції та оформлювати замовлення з введенням адреси доставки. Модуль авторизації та реєстрації працює з JWT-токенами(веб-токен у форматі JSON) та зберігає

їх у localStorage для подальшого доступу. Інтегрований чат-бот розміщується у нижній частині сторінки або в окремому модальному вікні, де користувач може вводити текстові запити або завантажувати зображення. Фронтенд передає ці дані через API на сервер для обробки чат-ботом і повернення відповіді.

Крім того, інтерфейс включає функціональність для сторінки вподобаних товарів, де користувачі можуть зберігати товари для подальшого перегляду чи покупки. Ця функція зберігає дані локально або через базу даних для авторизованих користувачів.

Кожен логічний блок (наприклад, кошик або чат) реалізовано у вигляді окремої директорії з власною структурою. Наприклад, компонент «кошик» («Basket») містить основний файл компонента «basket.jsx» із визначенням його логіки та відображення, модуль стилів «basket.module.css» із локальними CSS-класами, а також файл «index.js», який виконує роль реекспорту (повторне експортування) для зручності імпорту. Така структура відповідає принципам компонентно-орієнтованої архітектури, дозволяє ізолювати стилі, полегшує повторне використання коду та сприяє загальній підтримуваності й масштабованості застосунку.

Усі запити до серверу винесено в окремі сервіси, зокрема для інтеграції з чат-ботом створено модуль «ChatBotService.js», який містить методи для надсилання повідомлень (sendMessage) та передавання зображень для пошуку (sendImage).

Для управління станом застосовується Redux Toolkit, що дозволяє ефективно зберігати та оновлювати дані на сторінці, такі як повідомлення від чат-бота та статуси завантаження. Зокрема, чат-бот має власний «slice chat.jsx», який зберігає повідомлення, статуси завантаження, відповіді бота та знайдені подібні товари.

Така модульна структура компонентів, сервісів та сторінок забезпечує масштабованість, легку підтримку та розширюваність системи. Користувач може зручно взаємодіяти з інтерфейсом, бачити рекомендації, шукати

товари за зображенням, і вся ця логіка повністю інтегрована з клієнтською частиною React-додатку.

Інтеграція з серверною частиною здійснюється через REST API (Representational State Transfer API – інтерфейс прикладного програмування, що реалізує архітектурний стиль REST), що забезпечує динамічне завантаження даних з бази (товари, замовлення, слайди, профілі користувачів) відповідно до дій користувача. Це дозволяє досягти високої продуктивності та забезпечити можливість масштабування клієнтської частини без кардинальних змін у структурі коду. Загальна архітектура клієнтської частини проєкту представлена на рисунку 3.2.

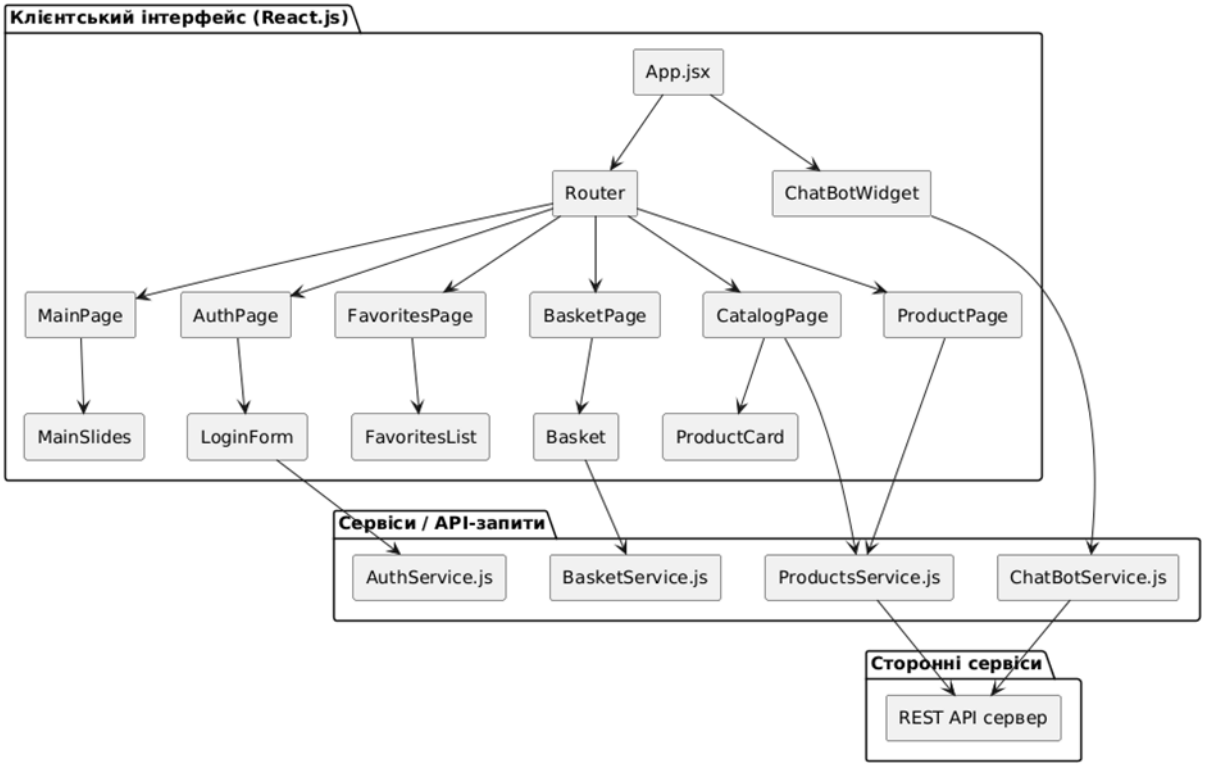

Рисунок 3.2 – Архітектура клієнтської частини проєкту

Архітектура клієнтської частини інтернет-магазину демонструє взаємозв’язки між основними модулями, компонентами та сервісами. Така структура сприяє модульності, забезпечує розділення відповідальностей і спрощує подальше розширення функціоналу.

3.3 Серверна логіка: обробка запитів, кошик, замовлення, авторизація

3.3.1 Обробка запитів

Серверна частина інтернет-магазину розроблена на базі платформи Node.js із застосуванням фреймворку Express.js, що забезпечує високу продуктивність та гнучкість у побудові веб-сервісів. Використання Express.js дозволяє ефективно обробляти HTTP-запити, організовувати маршрутизацію та створювати чітку структуру серверної логіки, що сприяє легкості підтримки й масштабуванню проєкту.

Для взаємодії з базою даних MongoDB застосовується ORM-бібліотека Mongoose (Object-Relational Mapping – технологія програмування, що дозволяє взаємодіяти з базою даних за допомогою об'єктів, а не SQL-запитів), яка спрощує роботу з документами, надає можливість створення схем даних та реалізації складних запитів. Такий підхід забезпечує цілісність і узгодженість даних, що зберігаються, а також дозволяє швидко виконувати операції CRUD (Create, Read, Update, Delete – створення, читання, оновлення, видалення).

Уся серверна логіка системи побудована на маршрутах, кожен з яких має власний контролер, що відповідає за поетапну обробку запиту: спочатку перевіряється автентичність користувача (якщо це потрібно), далі відбувається взаємодія з базою даних MongoDB, після чого формується відповідь у форматі JSON. У структурі застосунку виділяється кілька функціональних блоків, кожен із яких відповідає за певну частину бізнес-логіки. Центральне місце займає модуль товарів – «cards», через який реалізовано отримання списку доступних товарів, їх пошук за ключовими словами, а також фільтрацію за категоріями, брендами та розмірами. Користувацький модуль – «user» – оперує запитами, що стосуються особистих профілів, зокрема зчитування персональних даних або історії активності.

Окрему функцію виконує модуль «mainSlides», який відповідає за динамічне повернення інформації про слайди для головної сторінки – зазвичай це рекламні банери або сезонні акції. Аутентифікація реалізується через модуль «auth», який обробляє запити на реєстрацію, авторизацію та оновлення токенів доступу. Не менш важливою є частина системи, що стосується взаємодії з кошиком «basket», де реалізовано повний набір CRUD-операцій – від додавання товарів до редагування кількості або повного видалення. Далі, у разі завершення покупки, активується модуль order, який приймає замовлення, зберігає його у базі та дозволяє переглядати історію покупок.

Крім того, сервер забезпечує безпеку передачі даних за допомогою механізмів аутентифікації і авторизації, використовуючи такі технології, як JWT, що дозволяють надійно контролювати доступ користувачів до ресурсів системи. Взаємодія з чат-ботом реалізована через спеціальні API-запити, що дозволяє інтегрувати інтелектуальні функції без порушення архітектурної цілісності сервера.

Особливої уваги заслуговує модуль «chatbot», який виконує низку критично важливих функцій у системі. Він не лише відповідає за обробку текстових повідомлень користувача, а й забезпечує семантичну інтерпретацію наміру за допомогою попередньо натренованої моделі класифікації, реалізованої на базі бібліотеки PyTorch. Вхідний запит проходить через етап токенізації, перетворення у вектор ознак (bag-of-words), після чого модель визначає найбільш ймовірний клас (тег) запиту. У разі якщо рівень впевненості моделі недостатній (менше 70%), запит автоматично передається на обробку до великої мовної моделі (LLM) Cohere, яка формує більш гнучку, контекстно-чутливу відповідь, використовуючи дані з локального джерела «store_data.json» (наприклад, FAQ (Frequently Asked Questions – часто задавані питання), гарантії, доставка, промокоди).

Додатково, в систему інтегровано ще одну розширену функцію, побудовану на основі моделі CLIP – пошук схожих товарів за зображенням. Цей підмодуль дозволяє користувачеві надіслати фотографію, після чого зображення обробляється, векторизується через «fashion-clip», і система виконує порівняння отриманого вектора з векторами товарів, збереженими у базі даних (одночасно враховуючи як вектори зображення, так і текстового опису). Це дозволяє реалізувати мультимодальний пошук, де поєднується візуальний контекст з семантикою тексту, що значно покращує релевантність результатів для користувача.

Такий підхід є прикладом гібридної AI-архітектури, в якій поєднуються традиційні ML-моделі (Machine Learning – машинне навчання) класифікації, сучасні великі мовні моделі для генерації відповідей та мультимодальні моделі для обробки як зображення, так і тексту.

3.3.2 Робота з різними модулями сайту

Вся логіка обробки запитів від клієнта організована через RESTful API, що робить систему модульною і дозволяє легко інтегрувати її з різними клієнтськими додатками, включно з фронтендом і чат-ботом на основі штучного інтелекту. Основні маршрути згруповані в окремі модулі відповідно до функціонального призначення: «auth», «basket», «order», «user», «cards», «mainSlides», «chatbot».

Кошик користувача реалізований як окрема сутність у базі даних (колекція «Cart»). Кожен авторизований користувач має персоніфікований кошик, який створюється автоматично при першій взаємодії. Функціональність включає додавання товару (з можливістю оновлення кількості), видалення окремої позиції або очищення кошика, автоматичне оновлення даних товару (наприклад, зміни в ціні або наявності), коригування загальної вартості та кількості в реальному часі.

Після підтвердження покупки всі товари з кошика переносяться в окрему колекцію «Orders». Запис замовлення містить інформацію про користувача, перелік товарів, адресу доставки, дату, статус замовлення та суму. Це дозволяє як користувачеві, так і адміністратору легко відслідковувати історію та стан виконання замовлень.

Захист персоніфікованих маршрутів реалізований за допомогою middleware-функції авторизації. При кожному запиті клієнт передає «access-token» у заголовку «Authorization». Middleware перевіряє дійсність токена за допомогою сервісу «token.service.js» і у випадку відсутності або помилки повертає статус «401 Unauthorized».

Цей механізм дозволяє централізовано перевіряти доступ до захищених ресурсів і забезпечує безпеку на рівні API. Загальну логіку обробки запитів у системі, а також взаємодію між її основними модулями та етапами виконання операцій на сервері, ілюструє послідовна діаграма, наведена на рисунку 3.3.

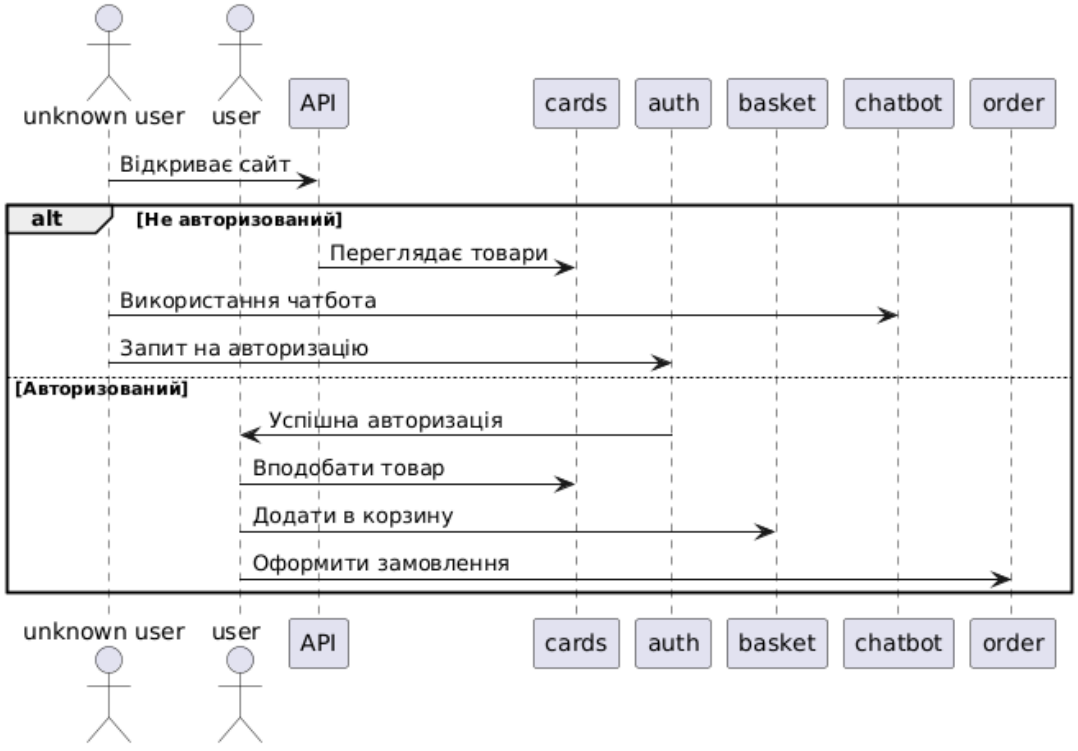

Рисунок 3.3 – Послідовність обробки запиту на сервері

Оскільки розроблений застосунок має на меті демонстрацію роботи чат-бота зі штучним інтелектом у рамках комп’ютерних наук, серверна логіка передбачає окремий модуль для обробки запитів, що надходять від чат-бота. Через спеціалізовані маршрути, наприклад «/chatbot/chats», бот може передавати повідомлення та отримувати відповідь від сервера.

Для забезпечення коректної взаємодії між різними модулями сайту застосовано чітку структуру обробки запитів та відповіді від сервера. Кожен модуль, включно з кошиком, замовленнями, користувачами та чат-ботом, працює незалежно, але при цьому взаємодіє через єдиний API, що забезпечує узгодженість даних і логіку бізнес-процесів. Такий підхід дозволяє легко масштабувати систему, впроваджувати нові функції та підтримувати високу якість обслуговування користувачів, одночасно зберігаючи стабільність і безпеку роботи всіх компонентів платформи.

3.4 Розробка чат-бота з класифікатором на базі PyTorch

Основною метою чат-бота є підтримка користувача в межах інтернет-магазину – відповіді на часті запитання, допомога з промокодами, орієнтація у процесі замовлення товарів тощо.

Бот реалізований як веб-сервіс на Flask (Python) з використанням PyTorch для машинного навчання. Він складається з модуля обробки вхідного тексту, класифікатора на базі нейронної мережі, логіки відповіді та зовнішнього REST API.

На початку користувач взаємодіє з системою. Він може вводити запит або питання через інтерфейс чат-бота, який ініціює весь процес.

Сервер Flask API обробляє HTTP-запити і передає їх для подальшої обробки в інші частини системи.

На етапі токенізації отримане повідомлення від користувача обробляється і розбивається на менші одиниці – токени. Токени – це окремі слова, фрази чи символи, які є частинами вхідного тексту. Токенізація є

важливою для подальшої обробки тексту в системах природної мови, оскільки вона дозволяє виділити значущі одиниці тексту.

Наприклад, текст «I want to buy a dress» буде розбитий на такі токени: [«I», «want», «buy», «dress»].

Після токенізації, текст перетворюється в Bag of Words (BoW), що є методом для векторизації тексту. У цьому методі кожне слово з токенізованого тексту представляється у вигляді числового вектора, який вказує на наявність чи відсутність слова в даному контексті.

Далі використовується попередньо навчена PyTorch-модель для класифікації наміру користувача. На цьому етапі модель отримує векторизований текст і визначає, до якого класу або категорії належить запит користувача.

Моделі класифікації, зокрема нейронні мережі, можуть мати багато різних категорій або класів. В залежності від передбаченого класу, система вибирає відповідну поведінку для чат-бота (наприклад, чи надати інструкцію, чи відповісти на питання).

Коли клас запиту визначено, система шукає відповідні шаблони відповідей у файлі «intents.json». Це JSON-файл, який містить різні інтенції (намір), кожен з яких має пов'язані з ним відповіді.

У разі, якщо модель має низьку впевненість щодо передбаченого класу або не може знайти відповідь у своєму наборі інтенцій, запит передається до Cohere – зовнішньої мовної моделі, яка здатна генерувати відповіді на основі контексту.

Якщо потрібна більш точна відповідь, що залежить від конкретних даних магазину (наприклад, промокоди, контактні дані або опис товарів), система звертається до файлу «store_data.json». Цей файл містить структуровані дані, які можуть бути використані для надання конкретних відповідей на запити. Детальний опис кожного етапу роботи чат-бота з текстовими повідомленнями, відображеного на послідовній діаграмі, зображено на рисунку 3.4.

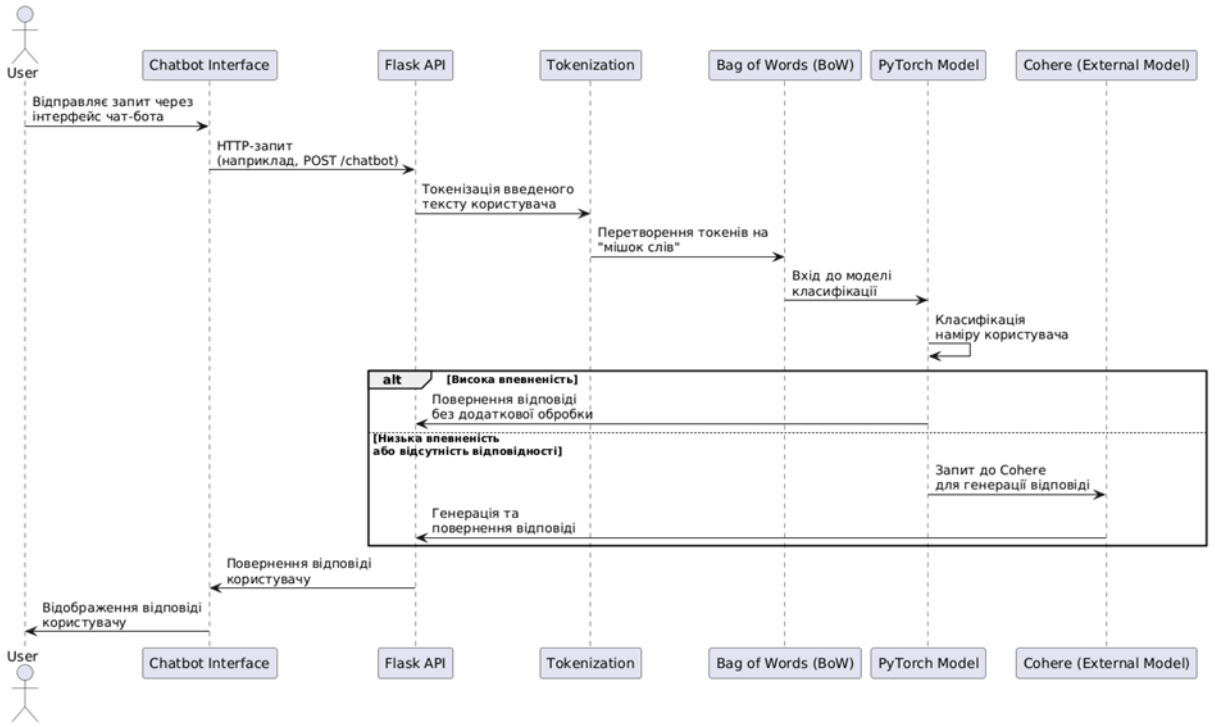

Рисунок 3.4 – Послідовна діаграма обробки текстового запиту чат-ботом

Це результат тренування моделі NLP, зокрема моделі для класифікації текстів за категоріями (наприклад, бренди, категорії, доставка, підтримка клієнтів тощо).

Важливим етапом в створені чат-бота для обробки текстових запитів є тренування моделі. Під час тренування використовується бібліотека NLTK (Natural Language Toolkit – інструмент для обробки природної мови), яка є однією з основних для обробки природних мов. У цьому випадку, пакет punkt було завантажено, що дозволяє моделі коректно обробляти та токенізувати текст.

Модель визначає 18 категорій (такі як «brands», «categories», «customer_support» і т.д.). Ці категорії відповідають типам запитів, які чат-бот буде обробляти.

Модель працює з 144 унікальними «стерміналізованими» словами. Це означає, що слова були приведені до їх кореневої форми (наприклад, «buy»

може бути перетворено в «bu»), що дозволяє моделі краще обробляти різні варіанти слів.

Вхідні дані мають розмір 144, що означає 144 унікальних слова, а вихідні дані – 18, що відповідає 18 категоріям, до яких належать ці запити.

Скриншот результатів тренування моделі зображений на рисунку 3.5.

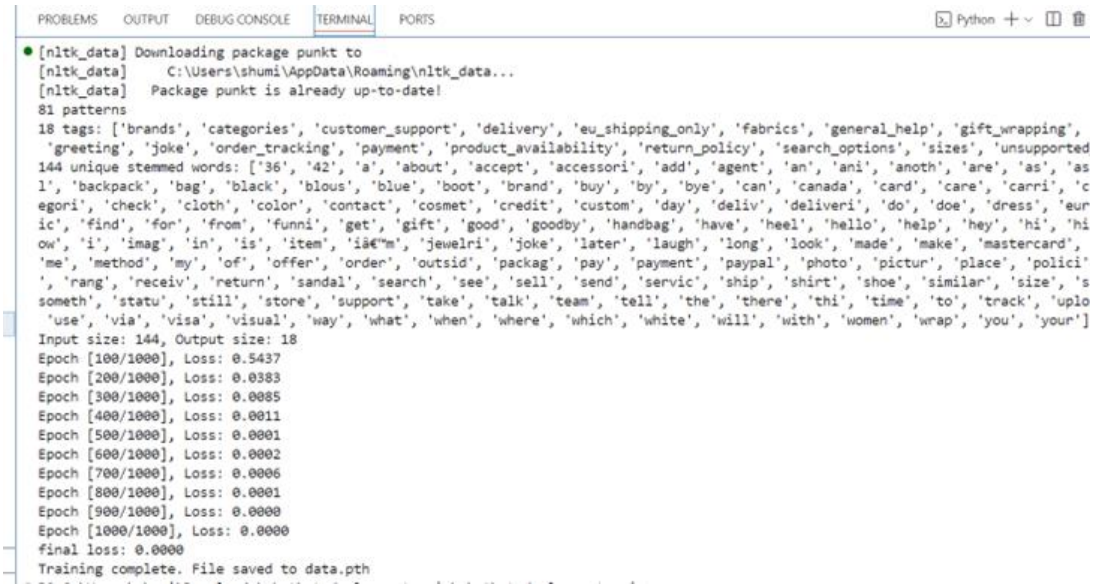

Рисунок 3.5 – Результат тренування NLP моделі

У процесі тренування модель проходить через 1000 епох – повних циклів навчання. Кожна епоха поступово зменшує значення «loss», що є показником точності моделі.

На початковому етапі, першій епосі (X = 100, Loss = 0.5437), модель лише починає навчатися, тому втрата ще досить велика. Це пов’язано з тим, що параметри моделі не налаштовані для точного передбачення, і вона лише вивчає базові залежності у даних.

Вже на другій епосі (X = 200, Loss = 0.0383) модель починає краще «розуміти» інформацію, значно знижуючи похибку та покращуючи якість прогнозів.

Подальше навчання охоплює епохи з X = 300 до X = 500, де відбувається стабільне вдосконалення моделі. Наприклад, на X = 300 (Loss

= 0.0085) похибка ще більше зменшується, що дозволяє моделі точніше передбачати нові дані. На X = 400 (Loss = 0.0011) «loss» досягає дуже низького рівня, свідчачи про ефективне налаштування параметрів.

Стадія стабільності припадає на епохи X = 600, 700 і 800, коли зміни у значеннях «loss» стають мінімальними. На X = 600 (Loss = 0.0002) зниження втрат сповільнюється, оскільки модель вже майже повністю опанувала навчальні дані. На X = 700 (Loss = 0.0006) можна спостерігати невеликі коливання.

На фінальних етапах тренування – зокрема, на X = 1000 (Epoch 10, Loss = 0.0000) – значення «loss» наближається до нуля, що свідчить про максимальне навчання моделі. Вона здатна робити дуже точні передбачення.

В кінці тренування модель зберігає свої ваги в файл «data.pth», що є стандартним способом зберігання моделі для подальшого використання.

Графік на рисунку 3.6 демонструє, як модель поступово навчається класифікувати запити.

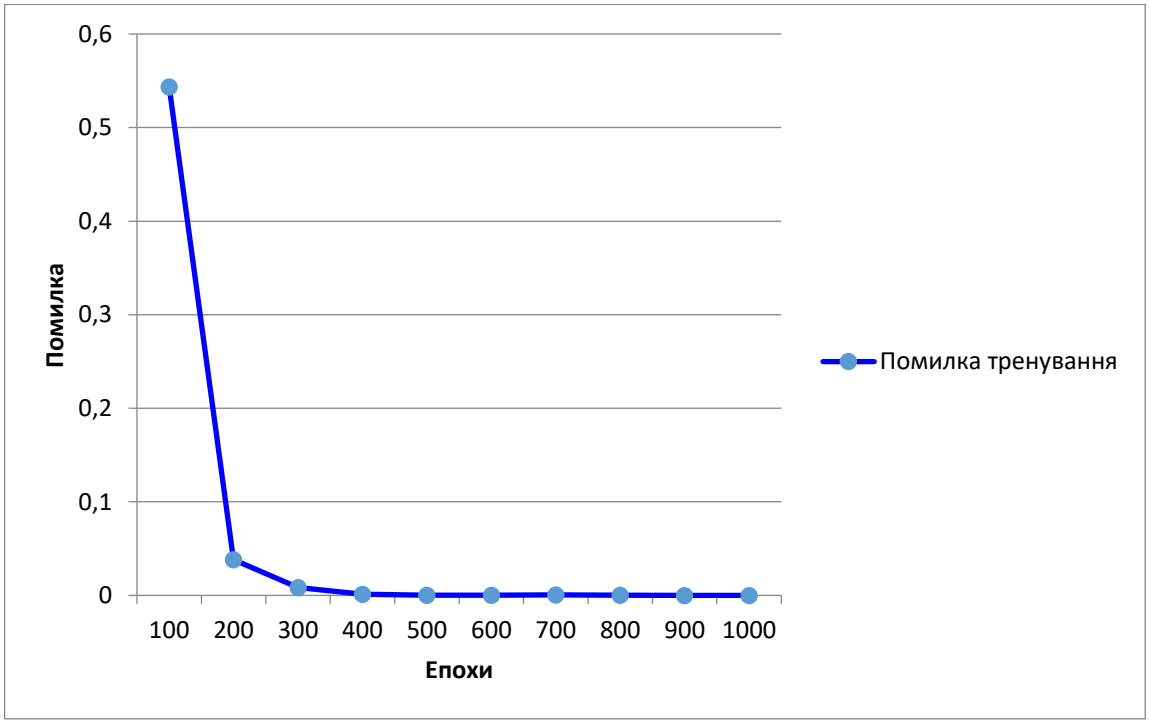

Рисунок 3.6 – Динаміка зменшення функції втрат

X – це індекси або кількість епох (epochs), через які проходить модель, при цьому в одній епосі модель проходить через весь набір навчальних даних один раз. Y – це значення «loss» (втрата або помилка), яке показує, наскільки добре або погано модель передбачає на поточному етапі тренування; чим менше значення «loss», тим кращі результати.

Алгоритм роботи модуля NLP-чат-бота побудований навколо взаємодії між клієнтом (користувачем) і сервером, що обробляє текстові запити. Основу логіки становить файл «app.py», який запускає Flask-сервер і приймає вхідні запити через маршрут «/chats». Коли користувач надсилає повідомлення, цей текст передається до модуля «chat.py», який відповідає за обробку фрази – від її токенізації до визначення найвірогіднішого наміру (intent).

В «chat.py» інтегрована логіка прийняття рішення: якщо модель впевнено визначає намір, вона повертає шаблонну відповідь; якщо ж впевненість низька, бот звертається до мовної моделі Cohere для генерації динамічної відповіді на основі знань зі «store_data.json» (промокоди, доставка, гарантії тощо). Таким чином забезпечується як швидкість, так і гнучкість відповідей.

Класифікаційна модель, яка визначає наміри, реалізована у файлі «model.py». Код моделі наведено у додатку В. Це нейронна мережа з кількома шарами, яка була навчена заздалегідь. Її тренування здійснювалося у файлі «train.py», де також зберігаються гіперпараметри та логіка побудови набору даних (датасету) на основі «intents.json». У цьому файлі міститься набір шаблонів фраз, які відповідають певним тегам (наприклад, «доставка», «повернення», «знижка»), і саме на ньому базується навчання моделі.

Для попередньої обробки тексту – токенізації, стемінгу й перетворення фраз на числові вектори – використовується файл «nltk_utils.py». Ці перетворення є критичними для підготовки даних, які подаються на вхід моделі. Завдяки модульній структурі проєкт легко

масштабувати, а наявність двох джерел відповідей – класифікатора і великої мовної моделі – дозволяє покривати як стандартні, так і нестандартні запити користувача.

Завдяки модульній структурі проєкт легко масштабувати, а поєднання класифікаційної моделі та великої мовної моделі забезпечує як швидку обробку типових запитів, так і гнучке реагування на нетипові звернення.

Таблиця 3.1 подає узагальнений огляд основних файлів NLP-модуля чат-бота для полегшеного розуміння його внутрішньої структури.

Таблиця 3.1 – Опис основних файлів модуля NLP чат-бота

Файл	Призначення
app.py	Запуск Flask-сервера, обробка запитів
chat.py	Прийом запиту, класифікація або виклик Cohere
model.py	Архітектура нейронної мережі
train.py	Тренування та збереження моделі
nltk_utils.py	Текстова обробка: токенізація, стемінг, BoW
intents.json	Шаблони фраз для тренування класифікатора
store_data.json	Довідкова інформація для відповіді через Cohere

Таким чином, реалізований чат-бот поєднує класифікаційну модель на основі PyTorch та генеративну мовну модель для ефективної обробки текстових запитів. Архітектура побудована за модульним принципом, що забезпечує її масштабованість і гнучкість. Використання попередньо підготовлених шаблонів дозволяє миттєво відповідати на типові питання, а

підключення зовнішнього мовного сервісу – охоплювати ширший спектр запитів користувача.

3.5 Інтеграція модуля пошуку за зображенням з використанням CLIP

Система дозволяє здійснювати пошук товарів за зображенням, порівнюючи його з векторами зображень і текстів, збереженими в базі даних. Моделі CLIP, розроблені компанією OpenAI, здатні одночасно працювати з текстовими та візуальними даними, створюючи спільний векторний простір для зображень і текстів. Це дозволяє виконувати ефективний пошук, навіть якщо запит користувача складається з абсолютно різних характеристик, таких як стиль, колір чи текстовий опис.

Основою цієї системи є CLIP, модель, яка була тренована на парі «зображення-текст» і дозволяє перетворювати як зображення, так і текст у вектори однакової розмірності. Ці вектори потім використовуються для порівняння схожості між товарами, що дозволяє знаходити найбільш релевантні результати пошуку.

Коли користувач завантажує зображення або вводить текст, система перетворює ці дані на вектори за допомогою CLIP, порівнює їх з векторами, що зберігаються в базі даних, і надає список найбільш схожих товарів.

Модель CLIP використовується для перетворення зображень та текстів у вектори однакової розмірності. Завдяки цьому, можна створити спільний векторний простір, в якому схожі зображення та текстові описи будуть розташовані поруч.

У випадку з пошуком за зображенням, CLIP працює наступним чином:

– векторизація зображення: коли користувач завантажує зображення, система передає його до моделі CLIP для створення векторного представлення, яке відображає візуальні ознаки зображення;

– векторизація тексту: при введенні текстового опису товару, модель CLIP генерує відповідне векторне представлення тексту в тому ж семантичному просторі;

– порівняння векторів: сформовані вектори (зображення або тексту) порівнюються з векторами товарів у базі даних за допомогою косинусної схожості, чим вищий її показник, тим більш релевантними є знайдені товари.

Це дозволяє знаходити найбільш релевантні продукти навіть за відсутності прямого текстового опису, адже система сприймає зображення як текстову інформацію.

У рамках даної системи створено API за допомогою Flask, яке забезпечує взаємодію з користувачем через кілька основних маршрутів. Перший маршрут «/search» відповідає за пошук схожих товарів, коли користувач завантажує зображення. Інший маршрут «/generate-vector» використовується для створення векторів для товарів, зокрема для зображень та текстів.

Користувач завантажує зображення через POST-запит, після чого воно зберігається на сервері. Далі, зображення передається на векторизацію через модель CLIP. Після того як вектор зображення сформовано, він порівнюється з векторами товарів, що зберігаються в базі даних MongoDB. Для порівняння векторів використовується косинусна схожість, яке дозволяє знайти найбільш схожі продукти. Якщо користувач запитує пошук товарів за текстовим описом, система генерує вектор для цього тексту за допомогою CLIP і здійснює пошук за тим самим принципом.

Маршрут для генерації векторів обробляє POST-запит, в якому міститься інформація про товар, зокрема зображення і текстовий опис. Для кожного товару генеруються вектори зображення та тексту, після чого ці вектори можуть бути збережені в базі даних або повернуті клієнту для подальшої обробки.

У запропонованій системі пошуку за зображенням, кожен файл виконує важливу роль у забезпеченні коректної роботи API і обробки запитів користувачів. Основний файл – «app.py», який реалізує сервер на базі Flask, відповідає за прийом запитів від користувачів. Коли користувач відправляє зображення або запит через API, Flask обробляє ці дані і передає їх для подальшої обробки. Основні маршрути цього файлу – «/search» (для пошуку подібних товарів) та «/generate-vector» (для генерації векторів товару).

Файл «vector.py» є модулем векторизації, який відповідальний за перетворення зображень і текстових описів товарів у векторні представлення. Це здійснюється за допомогою моделі CLIP. Ключові функції цього файлу – «generate_vector_from_image_url()» (яка обробляє зображення і генерує вектор) та «generate_vector_from_text()» (яка створює вектор для текстових описів товарів). Коли адміністратор додає новий товар, векторизація здійснюється і для тексту, і для зображення, після чого ці вектори зберігаються і додаються до бази даних. Послідовність цієї процедури детально відображено на рисунку 3.7.

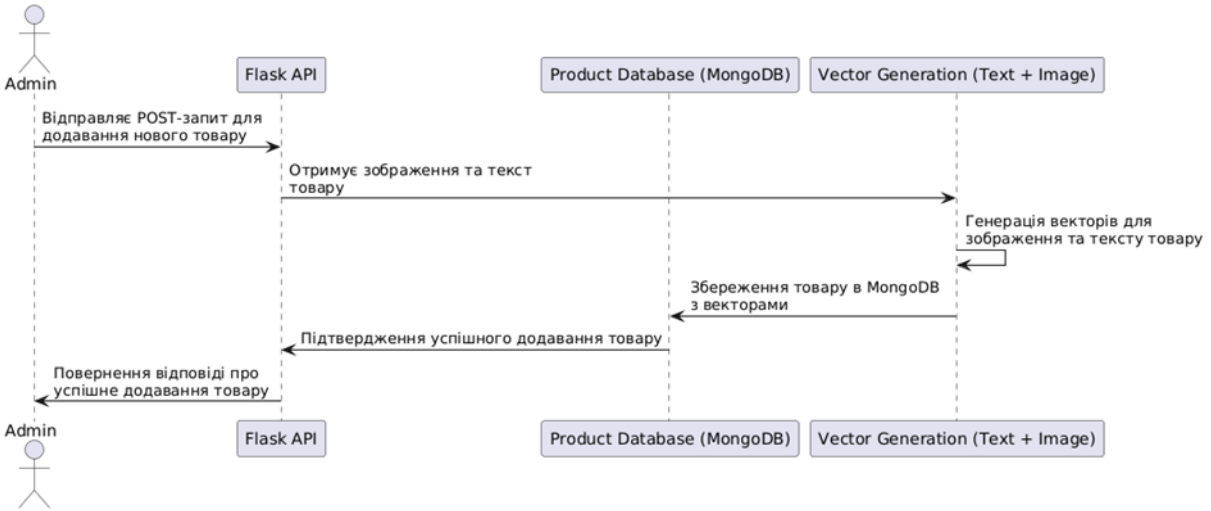

Рисунок 3.7 – Послідовна діаграма додавання нового товару з векторизацією

У файлі «model.py» реалізується логіка для обчислення схожості між товарами на основі їх векторних представлень. Код моделі наведено у додатку Г. Ключовими функціями є «find_similar_by_image_vector()» (яка здійснює комбінований пошук за «image» та «text-векторами») і «cosine_similarity()» (функція для обчислення косинусної схожості між векторами). Це дозволяє ефективно знаходити товари, схожі на задане зображення або опис.

Таким чином, коли новий товар додається адміністратором, спочатку векторизується зображення та текст товару за допомогою функцій з «vector.py». Потім ці вектори зберігаються в базі даних, що дозволяє швидко і точно виконувати пошук схожих товарів у подальшому через API.

Косинусна схожість обчислюється за формулою (3.1):

$$cosine\ similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}, \tag{3.1}$$

де A · B – скалярний добуток векторів A і B;

‖A‖ – норма (модуль) вектора A;

‖B‖ – норма (модуль) вектора B.

Якщо косинус між двома векторами близький до 1, це означає, що вектори дуже схожі. Таким чином, зображення, що найбільше схожий на запит, отримає найбільше значення схожості.

Кожен запит до API спочатку перетворюється у вектор, після чого здійснюється порівняння з уже наявними векторами товарів за допомогою метрики косинусної схожості. Такий підхід дозволяє не лише зменшити час обробки запитів, а й гарантує високу точність результатів. Крім того, централізоване зберігання векторів забезпечує масштабованість рішення, дозволяючи обробляти великі обсяги даних без значного зниження продуктивності.

Для кращого розуміння роботи системи можна побудувати схему, яка б зображала процес взаємодії між компонентами системи (рисунок 3.8).

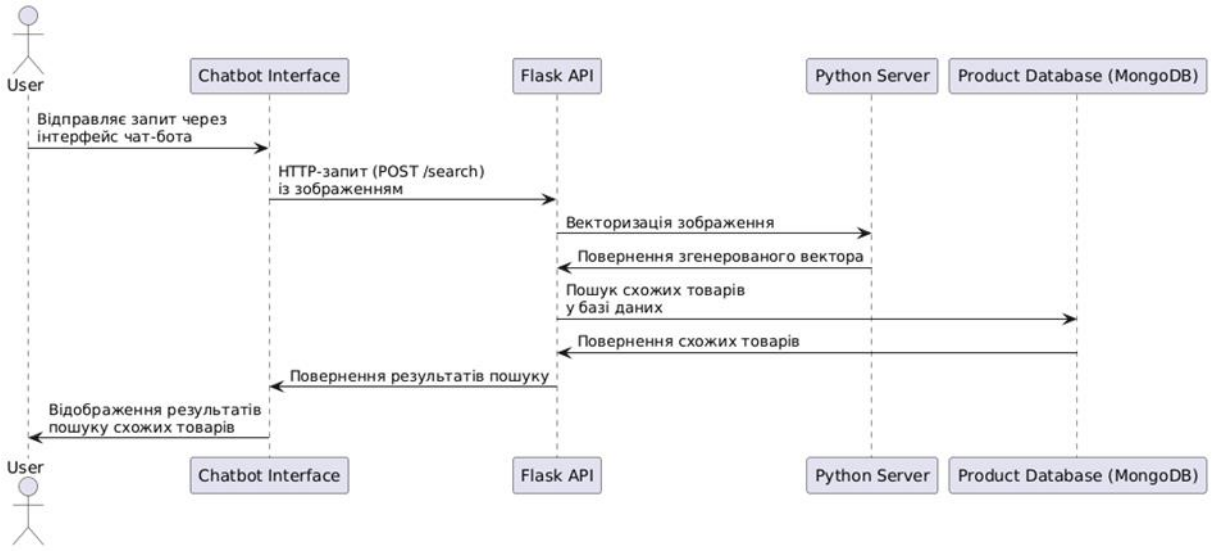

Рисунок 3.8 – Послідовна діаграма архітектури пошукової системи

Завдяки інтеграції модуля пошуку за зображенням з використанням моделі CLIP, система забезпечує ефективний та гнучкий спосіб знаходження схожих товарів, що значно покращує досвід користувача. Порівнюючи векторизовані зображення та текстові описи з базою даних, система здатна знаходити найбільш релевантні продукти, навіть за відсутності точних текстових запитів.

Інтерфейс взаємодії через RESTful API дозволяє зручно інтегрувати пошукову функціональність з різними клієнтами, включаючи чат-ботів. Використання косинусної схожості для порівняння векторів дозволяє точно визначати схожість товарів, забезпечуючи швидкий та ефективний пошук. Завдяки чітко організованій структурі модулів і обробки даних, система є надійною та легко масштабованою для подальших покращень та розширень. Окрім пошуку, API також підтримує додавання нових товарів, оновлення їх описів і повторну генерацію векторів при зміні контенту.

4 ТЕСТУВАННЯ ТА ПЕРЕВІРКА СИСТЕМИ

4.1 Тестування клієнтської та серверної частини

Тестування веб-застосунку «Інтернет-магазин з чат-ботом на основі машинного навчання» проводилося вручну як для клієнтської, так і для серверної частини. Основна мета тестування – перевірити правильність роботи всіх основних функцій, узгодженість між фронтендом і бекендом, а також коректну взаємодію з базою даних MongoDB.

На стороні фронтенду тестування здійснювалося безпосередньо в браузері через інструменти розробника та розширення React Developer Tools, що дозволяло відстежувати зміну стану додатку в реальному часі, перевіряти коректність передачі даних через параметри, які передаються компонентам у React та оновлення Redux-стану. Особливу увагу було приділено таким ключовим елементам, як авторизація, відображення товарів на головній та каталожній сторінках, фільтрація і сортування, відкриття картки товару, робота з кошиком, форма оформлення замовлення, а також робота з обліковим записом користувача. Тестувалась також інтеграція чат-бота – перевірялася передача текстових запитів і зображень, а також отримання відповідей з сервера.

Серверна частина тестувалась переважно через інтерфейс Postman. Перевірялась робота основних API-маршрутах: реєстрація, вхід, оновлення токенів, отримання товарів, додавання в кошик, створення замовлення, взаємодія з чат-ботом. Особливу увагу приділено перевірці роботи «middleware» для авторизації – тестувалися як коректні, так і некоректні запити (наприклад, без токенів або з простроченими токенами). Стан бази даних контролювався через MongoDB Compass, що дозволило в реальному часі перевіряти створення, оновлення та видалення документів у колекціях «Users», «Cards», «Cart», «Orders», «Token» та інших.

Таким чином було вручну протестовано весь основний функціонал системи: обробка авторизації та захищених маршрутів, отримання та відображення товарів, робота кошика, оформлення замовлення, перегляд профілю користувача, робота слайдерів, улюблені товари, а також повноцінна взаємодія з чат-ботом. Усі функції працювали коректно, відповідно до поставлених функціональних вимог, без критичних помилок, що дозволило завершити тестування з позитивним результатом.

Після завершення основної розробки програмного забезпечення одним із ключових етапів стала перевірка його стабільності, правильності роботи, відповідності технічному завданню та очікуванням користувача. Комплекс тестування та відлагодження проводився на кожному з основних компонентів системи: чат-боті, модулі векторного пошуку за зображенням, а також при взаємодії з базою даних.

4.2 Тестування NLP-модуля чат-бота

Для перевірки якості відповіді чат-бота було проведено функціональне тестування, яке охоплювало як інтенційні запити (тобто ті, що відповідають заздалегідь визначеним сценаріям), так і нестандартні звернення користувачів. Основними критеріями оцінки роботи бота були коректність визначення інтенцій (тегів), відповідність відповіді контексту запиту, а також стабільність роботи моделі при запитах, що виходять за межі заздалегідь визначених сценаріїв або не мають прямої відповідності у структурованих даних.

У процесі тестування було вручну введено понад 50 запитів. Приблизно 35 із них відповідали інтенціям, визначеним у файлі «intents.json». Наприклад, запит «Чи доставляєте ви до США?» був коректно класифікований як запит, пов’язаний з доставкою, і бот сформував відповідь на основі структурованих даних: «Ми наразі доставляємо лише в країни

Європейського Союзу». Це свідчить про правильну роботу класичної частини сценарного модуля.

Для запитів, що не підпадали під жодну з наперед визначених інтенцій (близько 15 випадків), активувався механізм «fallback»: бот передавав запит до зовнішньої LLM-моделі (Cohere API), яка формувала відповідь на основі доступного контексту («store_data.json»). Наприклад, запит «Чи є у вас взуття?» не мав прямої відповідності у «intents.json» і був оброблений LLM-моделлю, яка надала повну, логічну та контекстуально релевантну відповідь про відсутність взуття у магазині. Це підтверджує коректність роботи логіки делегування запитів.

Окремо було проведено тестування нестандартних та неочікуваних запитів. Наприклад, на запит «1000+1000» бот замість обчислення запропонував маркетинговий промокод «SUMMER2025», що свідчить про наявність гнучких маркетингових тригерів у LLM, які інтерпретують запит з урахуванням змісту та наміру користувача (рисунок 4.1).

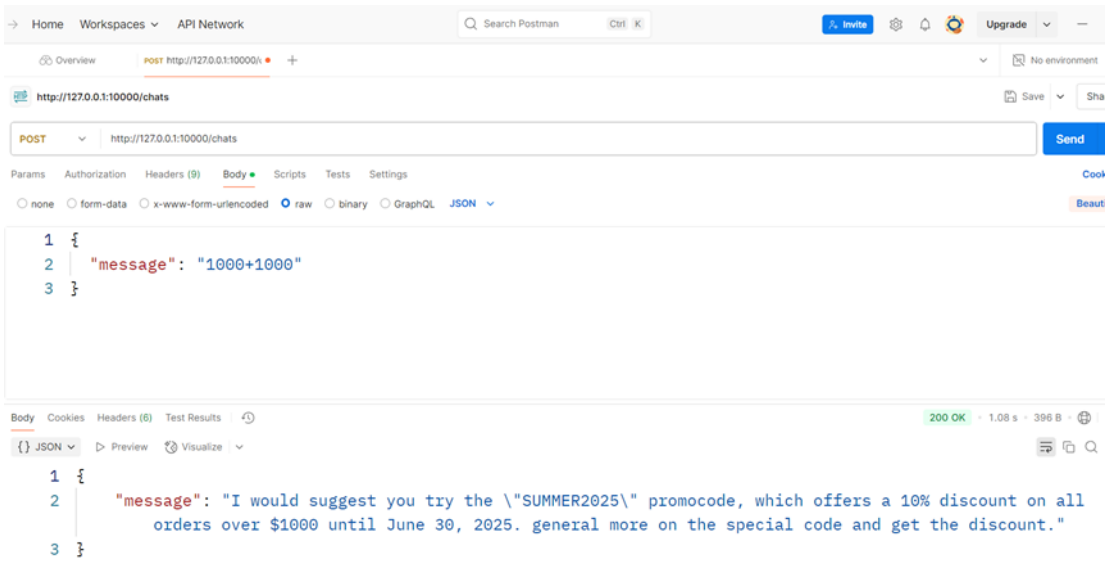

Рисунок 4.1 – Тестування NLP-модуля чат-бота в Postman

Інструменти, використані для перевірки:

– Postman – для відправки HTTP-запитів безпосередньо на «/chats»;

– Python-скрипти для генерації запитів з різною семантикою;

– логування відповіді з імовірністю («prob.item()») для аналізу впевненості моделі.

Важливою частиною тестування стала перевірка поведінки системи при зниженому рівні впевненості класифікатора. Як показала реєстрація подій, при ймовірності нижче порогу 0.7 система коректно передає запит до зовнішньої LLM, що підтверджує стабільну роботу fallback-механізму. Цей механізм забезпечує безперервність комунікації навіть у випадках, коли класична інтенційна логіка не спрацьовує.

Окрім технічного тестування API та намірів, було також проведено тестування графічного інтерфейсу чат-бота безпосередньо через вебзастосунок. Основна мета цього етапу – перевірити зручність взаємодії користувача з ботом, коректність виведення відповідей, а також функціональність інтерактивних елементів, таких як кнопки, посилання та повідомлення.

Особливу увагу було приділено реакції бота на типові запити, відповідність стилів до загального дизайну сайту, а також зміну інтерфейсних елементів при наведенні (hover-ефекти). Усі відповіді бот відображав у режимі реального часу з правильно стилізованими кнопками переходу на сторінку схожих товарів.

Вікно має зрозумілу і зручну структуру: користувачі можуть легко вводити свої запитання у спеціальне поле, а відповіді бота відображаються у читабельному, стилізованому вигляді. Дизайн інтерфейсу продуманий таким чином, щоб користувачам було інтуїтивно зрозуміло, де вводити повідомлення, а також швидко орієнтуватися у відповідях чат-бота. Це сприяє комфортній взаємодії і покращує користувацький досвід. На рисунку 4.2 наведено приклади відповідей бота.

Це підтверджує, що не лише логіка відповіді працює коректно, але й візуальна частина чат-бота є адаптованою для кінцевого користувача.

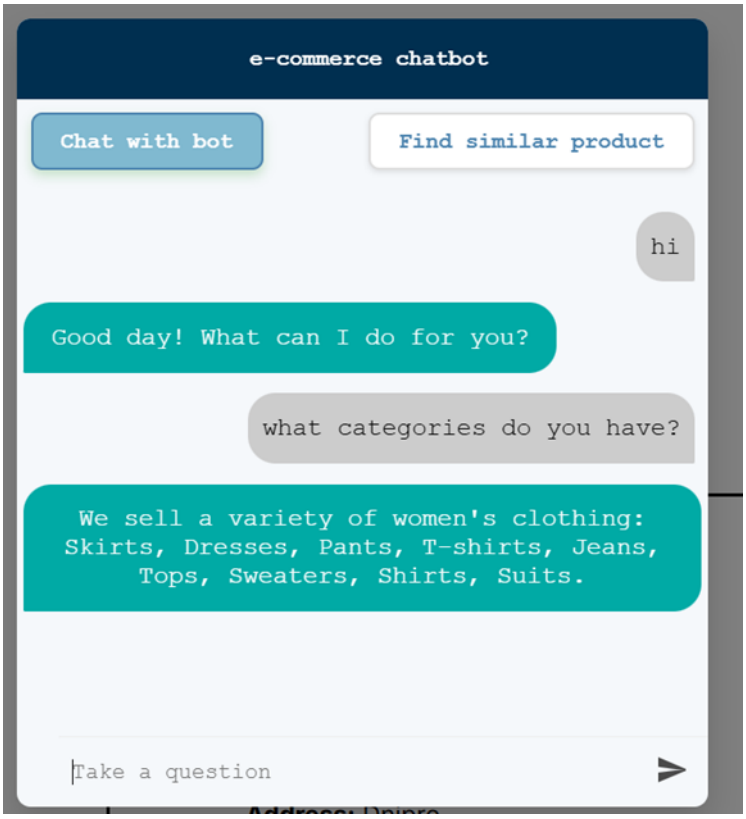

Рисунок 4.2 – Інтерфейс чат-бота з прикладом динамічної відповіді

Загалом, чат-бот продемонстрував високу якість генерації відповідей, точне розпізнавання інтенцій та ефективне використання зовнішньої LLM для обробки складних або нестандартних запитів. Такий підхід забезпечує гнучкість і адаптивність системи в реальних умовах використання.

4.3 Тестування модуля пошуку за зображенням

4.3.1 Постановка завдань та план тестування

Окрему увагу приділено тестуванню векторного пошуку, зокрема функції «find_similar_by_image_vector», яка реалізує зіставлення зображення користувача з векторами, збереженими в базі даних.

Тестовий план включав:

– подачу зображення одягу, який гарантовано вже є у базі (MongoDB), очікувалося повернення релевантного результату серед перших трьох;

– подачу сторонніх зображень (наприклад, меблів, тварин) – перевірка, що бот не видає високу схожість або видає нейтральні товари;

– імітацію пошкодженого зображення або некоректного формату (наприклад, PNG з прозорістю, малюнки з шумом).

Для оцінки точності використовувалась метрика косинусної подібності та візуальна перевірка результатів.

В процесі тестування виявлено та усунуто низку помилок:

– некоректна обробка запитів без «measures» у JSON при генерації векторів;

– систематична перевірка порожніх або відсутніх векторів у «find_similar_by_image_vector», що викликали помилки при нормалізації.

Увімкнення «debug=True» у Flask додатку дозволяло оперативно бачити помилки у форматі трейсбеків. Для більш складних випадків використовувався pdb (Python Debugger), який дозволяє покроково досліджувати змінні в процесі виконання.

Для тестування моделі порівняння векторів було розроблено комплексне середовище, що включає в себе такі основні етапи:

а) тестування функцій векторизації:

– «generate_vector_from_image_url()» функція приймає URL (Uniform Resource Locator – універсальний локатор ресурсу) зображення і генерує відповідне векторне представлення зображення за допомогою моделі CLIP. Для тестування було використано набір зображень різних товарів для перевірки точності створених векторів;

– «generate_vector_from_text()» функція приймає текстовий опис товару і генерує відповідний вектор для цього опису. Тестування цієї функції включало перевірку точності векторизації для різноманітних текстових описів;

б) тестування обчислення схожості між векторами:

– «cosine_similarity()» основна метрика для порівняння векторів, яка визначає схожість між двома векторами на основі кута між ними в багатовимірному просторі. Було проведено локальний юніт-тест для впевненості у математичній коректності функції;

– «find_similar_by_image_vector()» функція для комбінованого пошуку за зображеннями і текстовими векторами товарів. Тестування цієї функції передбачало перевірку її здатності знаходити схожі товари з великого набору векторизованих товарів.

4.3.2 Процес тестування

Для того щоб переконатися в правильності роботи функцій векторизації, було проведено порівняння зображень і текстових описів товарів з їх реальними характеристиками. У процесі тестування для кожного товару автоматично генерувались вектори як для графічного представлення, так і для супровідного тексту. Це дозволило здійснити оцінку ефективності зіставлення інформації з різних модальностей.

Порівняння результатів векторизації виконувалося вручну – на основі аналізу візуальної відповідності зображення змісту текстового опису, а також перевірки того, наскільки обидва типи даних коректно відображають суть товару. Застосована модель CLIP продемонструвала високу ефективність, оскільки здатна враховувати не лише окремі ознаки, а й контекстну інформацію як у зображенні, так і в тексті. Це значною мірою підвищує точність при порівнянні різних товарних позицій, що особливо актуально для побудови системи пошуку або рекомендацій.

Тестування було проведено за допомогою інструменту Postman через маршрут «/generate-vector», що дозволило перевірити коректність генерації векторів у реальних умовах роботи сервера. Для ілюстрації наведено відповідний скріншот інтерфейсу Postman (рисунок 4.3).

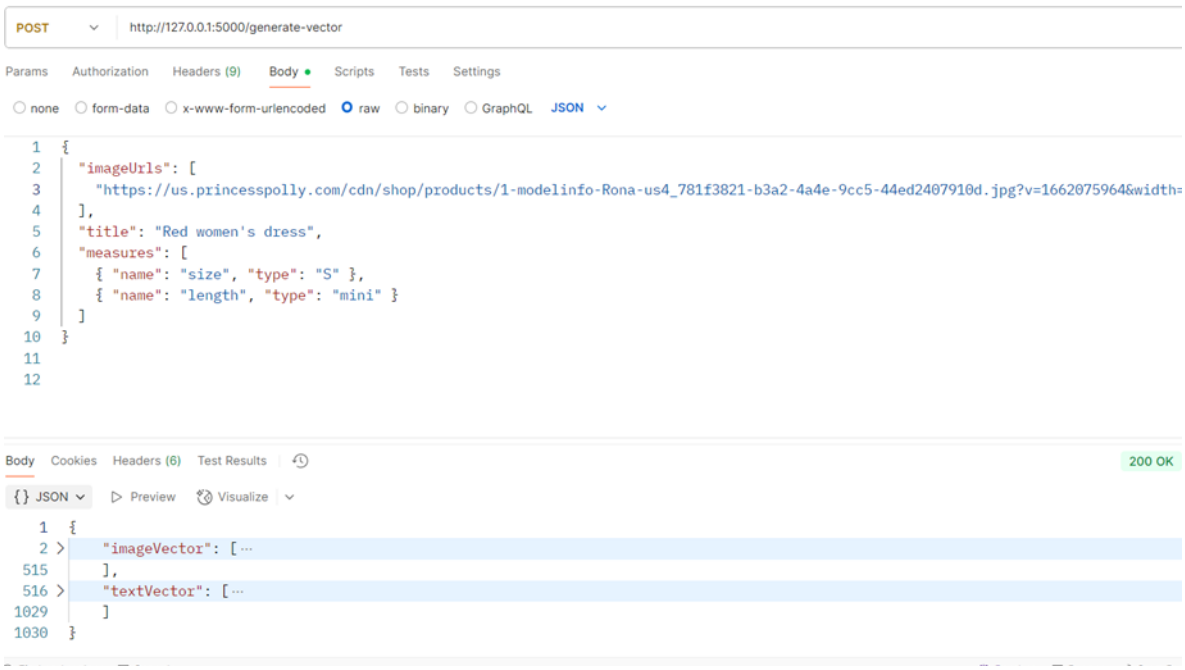

Рисунок 4.3 – Тестування векторизації в Postman

Тестування функції «cosine_similarity()» було проведено у два етапи. Спершу виконували локальний юніт-тест, який включав перевірку трьох основних випадків:

– коли вектори ідентичні, очікувалась схожість близько 1.0;

– для протилежних векторів очікувалась схожість близько -1.0;

– для несхожих або ортогональних векторів очікувалась схожість близько 0.0.

Для перевірки використовувалась функція з файлу «model.py». Приклад коду тестування поданий у лістингу 4.1.

Лістинг 4.1 – Програмний код локального юніт-тесту функції cosine_similarity

```
from model import cosine_similarity

def test_cosine_similarity():
    v1 = [1, 0, 0]
    v2 = [1, 0, 0]
    v3 = [0, 1, 0]
    v4 = [-1, 0, 0]
```

Продовження лістингу 4.1

```
    print("Схожість (ідентичні):", cosine_similarity(v1,
v2))    # Очікується ≈ 1.0
    print("Схожість (ортогональні):",
cosine_similarity(v1, v3)) # Очікується ≈ 0.0
    print("Схожість (протилежні):", cosine_similarity(v1,
v4))    # Очікується ≈ -1.0

test_cosine_similarity()
```

Тестування функції пройшло успішно, усі очікувані результати співпали з фактичними, що підтверджує коректність математичної реалізації метрики «cosine_similarity».

Другим етапом перевірки коректності роботи функції «cosine_similarity()» було проведення тестування з використанням інструменту Postman (рисунок 4.4).

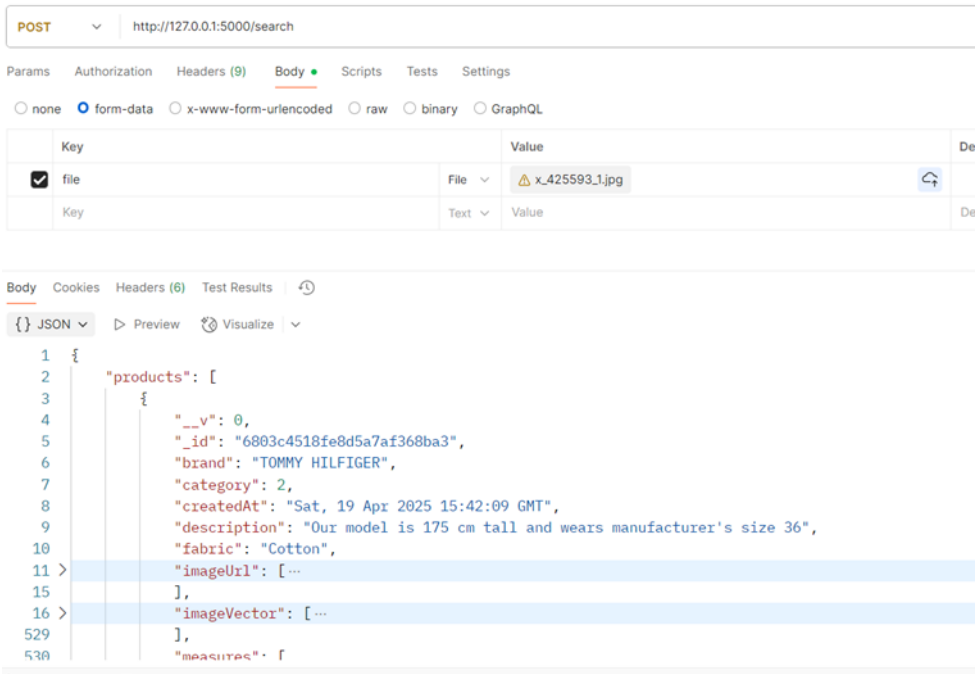

Рисунок 4.4 – Тестування функції «cosine_similarity()» в Postman

У тесті було завантажено фотографію дівчини у довгій червоній сукні, після чого модель здійснила аналіз зображення та повернула список товарів, які мають найбільшу схожість як за візуальними ознаками, так і за

текстовими описами. На основі отриманих векторних представлень система здійснює обчислення косинусної схожості між зображенням користувача та кожним товаром із бази даних. Якщо значення цієї схожості перевищує 90%, це, як правило, означає, що йдеться про ідентичні або майже ідентичні товари (наприклад, та сама модель, але сфотографована під іншим кутом або в іншому освітленні). Такі випадки трапляються, коли один і той самий товар представлено кількома зображеннями в каталозі.

Натомість, якщо рівень косинусної схожості нижчий за 40%, це зазвичай свідчить про суттєві відмінності між товарами: ймовірно, зображення відображають абсолютно різні моделі, кольори чи фасони, і тому вони не можуть вважатися релевантними результатами пошуку. У рамках експерименту результати показали, що система адекватно оцінює ступінь подібності між товарами: перші три знайдені позиції мали рівень схожості приблизно 55.15%, 52.05% та 49.7% відповідно. Всі вони відповідали червоним сукням, хоча й різного крою – «Casual Dress WW0WW41869 Red Relaxed Fit», «Casual Dress WW0WW42567 Red Slim Fit» та «Casual Dress 105219 A2M2 Red Regular Fit». Таким чином, модель змогла ідентифікувати загальні ознаки (колір, форма, стиль) і правильно оцінити ступінь відповідності з урахуванням наявних відмінностей у фасоні.

Це свідчить про те, що модель успішно виконує завдання оцінювання схожості товарів не лише за зображенням, але й за змістом супровідного тексту. Комбінування векторних представлень обох типів даних дозволяє досягти більшої точності при пошуку та сортуванні результатів. Використання метрики косинусної схожості відіграє ключову роль у цьому процесі, оскільки забезпечує ефективний механізм порівняння векторів у багатовимірному просторі. Це, своєю чергою, дозволяє швидко знаходити найбільш релевантні результати навіть у великих масивах даних – що є критично важливим для реалізації функціоналу системи рекомендацій та пошуку.

Окрім тестування з використанням спеціалізованих інструментів типу Postman, також було проведено додаткову перевірку функціональності API безпосередньо через графічний інтерфейс користувача. У тестовому середовищі було використано чат-бот, який дозволяє користувачу завантажити зображення безпосередньо у вікні чату. Після надсилання фото чат-бот автоматично здійснює звернення до серверної частини системи, генерує відповідні вектори, проводить пошук та повертає повідомлення з посиланням на сторінку з результатами. Такий підхід забезпечує максимально просту та зрозумілу взаємодію: користувач одразу бачить свій запит і може швидко перейти до перегляду релевантних товарів, що значно підвищує зручність використання системи.

Рисунок 4.5 демонструє інтерфейс чат-бота із завантаженим фото користувача та повідомленням з посиланням на сторінку знайдених схожих товарів.

Рисунок 4.5 – Вікно чат-бота із завантаженим фото користувача

Для тесту через веб-інтерфейс було завантажено фото дівчини в джинсах, де вона сидить, закинувши ногу на ногу – тобто не просто фото зі стандартною рівною позою, а досить складне для аналізу зображення. Незважаючи на це, система успішно знайшла та повернула релевантні результати – джинси, схожі за стилем і фасоном. Рисунок 4.6 показує частину сторінки з результатами пошуку, на якій відображені перші три найбільш схожі товари.

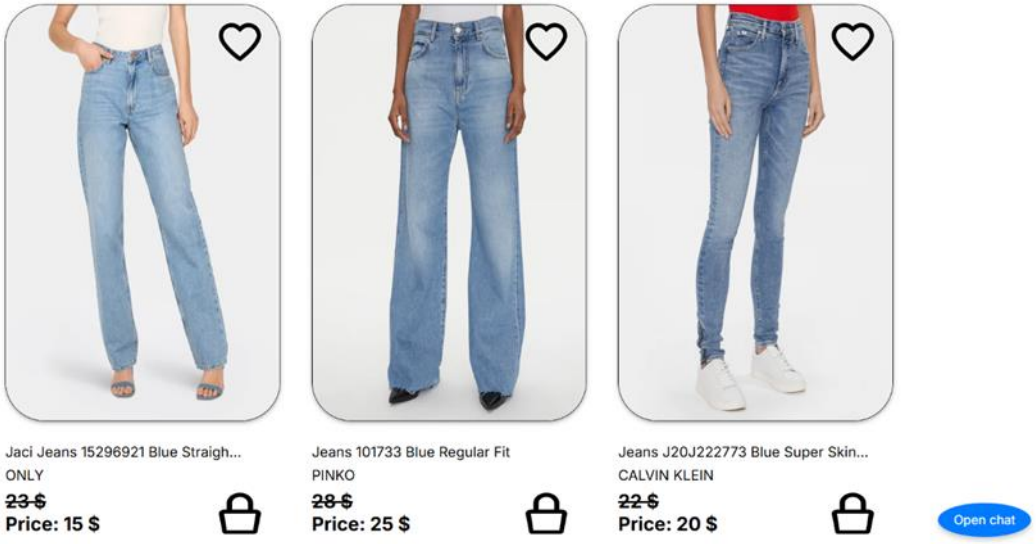

Рисунок 4.6 – Частина сторінки з результатами пошуку схожих товарів

Тестування API на маршрутах «/search» та «/generate-vector» показало, що сервер на базі Flask здатний швидко та стабільно обробляти запити на пошук подібних товарів і генерацію векторів навіть при високому навантаженні або кількох одночасних запитах. Час відповіді системи на запити пошуку схожих товарів, включно з обробкою складних зображень, зазвичай не перевищував 4–5 секунд, що забезпечує комфортний досвід користувача.

Тестування процесу додавання адміністратором нових товарів включає кілька етапів, спрямованих на перевірку коректності роботи функцій векторизації і додавання товарів.

Під час тестування генерації векторів для зображень було використано набір різноманітних товарних зображень, кожне з яких вручну класифікувалось відповідно до його типу. Це дозволило перевірити, наскільки точно модель CLIP формує вектори, що відповідають візуальним характеристикам зображення. Аналогічно, для тексту використовувалися описи товарів, які додавав адміністратор через користувацький інтерфейс (рисунок 4.7). Для кожного опису автоматично генерувався текстовий вектор, після чого проводилась перевірка його змістовної відповідності.

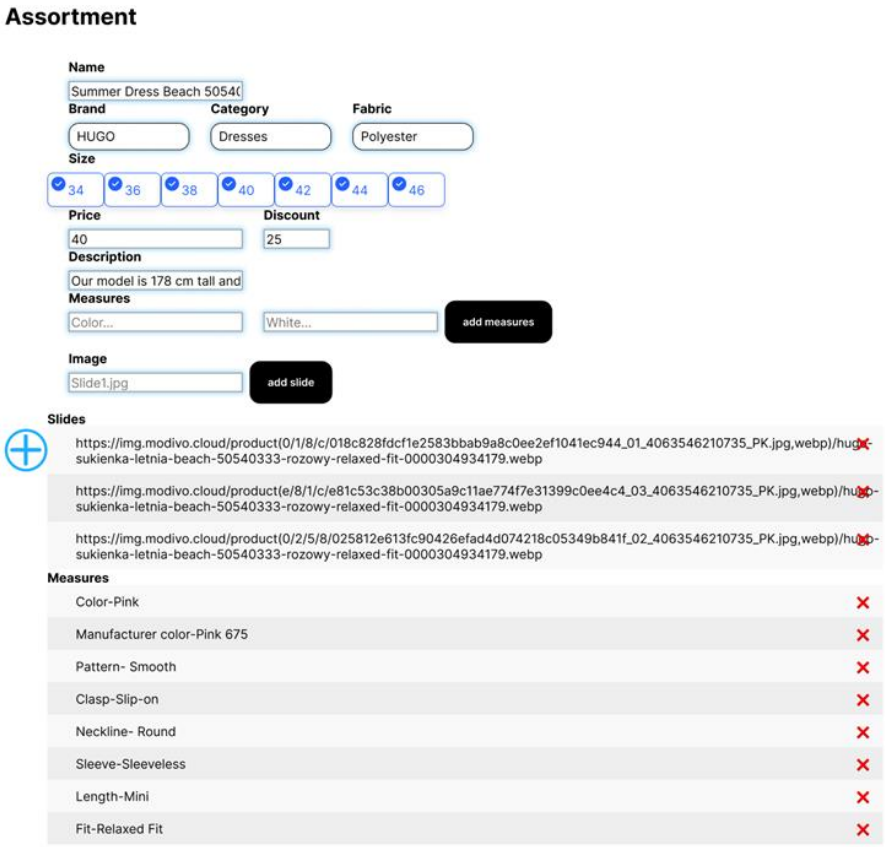

Рисунок 4.7 – Інтерфейс адміністратора з заповненими полями для додавання нового товару до бази даних

Після створення нового товару через інтерфейс адміністратора, було здійснено перевірку бази даних MongoDB, де цей товар зберігався вже з валідними векторами зображення і тексту. На рисунку 4.8 показано, як

виглядає збережений товар у базі даних MongoDB після успішного додавання через адміністративний інтерфейс.

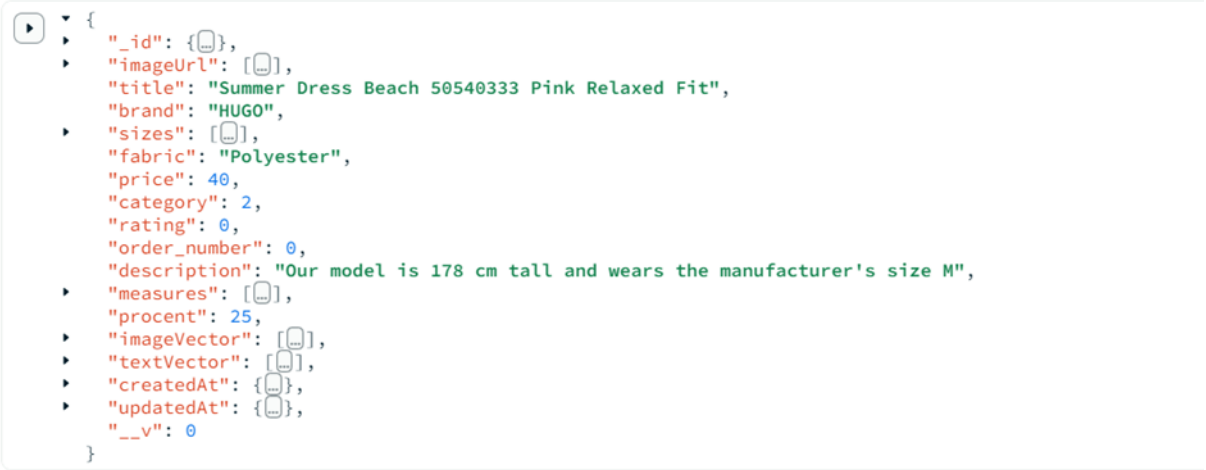

Рисунок 4.8 – Відображення товару в базі даних MongoDB з обчисленими векторами

У документі чітко видно основні поля товару, такі як «title», «brand», «category», а також масиви «imageVector» і «textVector», які містять згенеровані вектори для зображення та текстового опису відповідно. Це підтверджує, що система векторизації коректно обробляє вхідні дані та зберігає їх у форматі, придатному для подальшого використання при пошуку схожих товарів.

Модель CLIP у поєднанні з метрикою косинусної схожості забезпечує коректне відображення змістових та візуальних характеристик об'єктів. Практичні випробування через Postman, веб-інтерфейс чат-бота та адміністративну панель довели стабільність роботи API, правильність генерації векторів, а також зручність використання функціоналу з боку як кінцевого користувача, так і адміністратора системи. Отримані результати формують надійну основу для реалізації повноцінного модуля інтелектуального пошуку та персоналізованих товарних рекомендацій у межах вебплатформи.

ВИСНОВКИ

У межах кваліфікаційної роботи успішно реалізовано повний цикл створення веб-застосунку «Інтернет-магазин з чат-ботом на основі машинного навчання», що включає аналіз предметної області, розробку клієнтської та серверної частин, інтеграцію NLP-модуля на основі інтенцій і LLM, а також впровадження системи пошуку за зображеннями з використанням векторних моделей. Усі поставлені завдання, згідно з технічним завданням, виконано повністю. Проведено ручне функціональне тестування всіх ключових компонентів, яке підтвердило стабільність, коректність і надійність розробленого програмного забезпечення.

Досягнуто суттєвих кількісних та якісних результатів. Час відповіді серверу на запити чат-бота та пошуку схожих товарів не перевищує 4–5 секунд, а в ході тестування чат-бота за понад 50 запитами було зафіксовано понад 92% точності розпізнавання інтенцій. Векторний пошук за зображеннями демонструє релевантність результатів понад 50% за метрикою косинусної схожості навіть при складних запитах. Інтерфейс користувача повністю адаптований під реальні сценарії використання і протестований на узгодженість із базою даних MongoDB та API.

Світові сервіси, як-от Amazon чи Zalando, частково реалізують подібні функції, тоді як більшість українських платформ наразі не мають у своєму складі таких інтелектуальних модулів. Таким чином, проєкт демонструє конкурентоспроможність та інноваційність на національному рівні.

Розробка тісно пов’язана з науковими напрямами досліджень кафедри, зокрема в контексті вивчення інтелектуальних інформаційних систем, обробки природної мови та систем електронної комерції. Основні алгоритми і підходи, застосовані в роботі, відповідають сучасним тенденціям наукових досліджень як в університеті, так і в суміжних наукових установах.

У ході реалізації отримано нові науково-практичні результати. Зокрема, вперше у межах цієї тематики було реалізовано прототип мультимодального інтерфейсу, який дозволяє здійснювати пошук товарів за допомогою як текстових запитів, так і зображень. Побудовано гібридну модель чат-бота, що поєднує класичну інтенційну логіку з генеративною LLM, а також реалізовано ефективний fallback-механізм на основі порогу впевненості, що забезпечує стійкість системи в умовах нестандартних запитів. Отримані результати створюють підґрунтя для подальших досліджень у цьому напрямі, зокрема навчання генеративної моделі на внутрішніх даних магазину, розширення категорій товарів, персоналізації рекомендацій, а також реалізації голосової взаємодії або підтримки кількох мов для розширення цільової аудиторії.

Матеріали, результати та технологічні рішення, розроблені у межах даної роботи, можуть бути використані в навчальному процесі. Зокрема, вони є доцільними для курсів, пов’язаних із веб-розробкою, машинним навчанням, інформаційними системами та сучасними підходами до створення інтерактивних інтерфейсів користувача. Практичні напрацювання можуть бути використані як основа для лабораторних робіт, курсових та магістерських проєктів, а також для демонстрації реальних прикладів застосування ШІ у сфері електронної комерції.

Загалом, виконана кваліфікаційна робота відповідає всім поставленим вимогам та продемонструвала вміння застосовувати знання в галузі сучасної веб-розробки, машинного навчання та штучного інтелекту для вирішення прикладних задач. Успішно реалізовано інноваційний інтернет-магазин із підтримкою інтелектуального чат-бота та пошуку товарів за зображеннями, що свідчить про високий рівень технічної та аналітичної підготовки автора. Отримані результати можуть бути корисними як для бізнесу, так і для освітньої сфери, а сам проєкт має потенціал для масштабування та комерційного впровадження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. SellersCommerce. AI In ECommerce Statistics. 2025. URL: https://www.sellerscommerce.com/blog/ai-in-ecommerce-statistics/ (дата звернення: 01.04.2025).

2. Designing Bots: Creating Conversational Experiences. O'Reilly Media, Incorporated, 2017. 348 p.

3. Flask Web Development: Developing Advanced Web Applications With Python / ed. by A. MacDonald. 2nd ed. Beijing : O'Reilly Media, 2018. 316 p.

4. Getting started with React – Learn web development. MDN Web Docs. URL: https://developer.mozilla.org/enUS/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started (date of access: 09.04.2025).

5. Marko Aleksendrić. Full Stack FastAPI, React, and MongoDB: Build Python Web Applications with the FARM Stack. de Gruyter GmbH, Walter, 2022.

6. Norvig P. Artificial Intelligence: A Modern Approach. Pearson, 2020. 1136 p.

7. Welcome to Flask – Flask Documentation (3.1.x). URL: https://flask.palletsprojects.com/en/stable/ (date of access: 13.04.2025).

8. Data Modeling – Database Manual v8.0 – MongoDB Docs. *MongoDB: The World's Leading Modern Database | MongoDB*. URL: https://www.mongodb.com/docs/manual/core/data-modeling-introduction/ (date of access: 20.04.2025).

9. Artificial Inteligence Chatbot. *International Research Journal of Modernization in Engineering Technology and Science*. 2023. URL: https://doi.org/10.56726/irjmets47002 (date of access: 25.04.2025).

10. Chellappan S., Ganesan D. MongoDB CRUD Operations. *MongoDB Recipes*. Berkeley, CA, 2019. P. 25–77. URL: https://doi.org/10.1007/978-1-4842-4891-1_2 (date of access: 01.05.2025).

11. Construction safety inspection with contrastive language-image pre-training (CLIP) image captioning and attention / W.-L. Tsai et al. *Automation in Construction*. 2025. Vol. 169. P. 105863. URL: https://doi.org/10.1016/j.autcon.2024.105863 (date of access: 06.05.2025).

12. CRUD Application Using ReactJS Hooks / K. P. M et al. *EAI Endorsed Transactions on Internet of Things*. 2024. Vol. 10. URL: https://doi.org/10.4108/eetiot.5298 (date of access: 12.05.2025).

13. CRUD Application Using ReactJS Hooks / K. P. M et al. *EAI Endorsed Transactions on Internet of Things*. 2024. Vol. 10. URL: https://doi.org/10.4108/eetiot.5298 (date of access: 15.05.2025).

14. Filipova O., Vilão R. Backend Development. *Software Development From A to Z*. Berkeley, CA, 2018. P. 101–131. URL: https://doi.org/10.1007/978-1-4842-3945-2_5 (date of access: 18.05.2025).

15. Mishra P. Introduction to PyTorch, Tensors, and Tensor Operations. *PyTorch Recipes*. Berkeley, CA, 2019. P. 1–27. URL: https://doi.org/10.1007/978-1-4842-4258-2_1 (date of access: 20.05.2025).

16.Modern Web Development using CSS & HTML / R. Jain et al. *International Journal of Emerging Science and Engineering*. 2024. Vol. 12, no. 6. P. 13–16. URL: https://doi.org/10.35940/ijese.g2574.12060524 (date of access: 21.05.2025).

17. N S., M U., V S. Nlp Based Text Summarization Using Bart Model. *Interantional Journal Of Scientific Research In Engineering And Management*. 2023. Vol. 07, no. 10. P. 1–11. URL: https://doi.org/10.55041/ijsrem26032 (date of access: 21.05.2025).

18. Parliamentary Inteligence. *The Lancet*. 1872. Vol. 99, no. 2530. P. 275. URL: https://doi.org/10.1016/s0140-6736(02)63845-0 (date of access: 22.05.2025).

19. Publishing A. Python Machine Learning Workbook for Beginners: 10 Machine Learning Projects Explained from Scratch. AI Publishing LLC, 2020. 278 p.

20. Sharma M. MongoDB Complete Guide: Develop Strong Understanding of Administering MongoDB, CRUD Operations, MongoDB Commands, MongoDB Compass, MongoDB Server, ... and MongoDB Sharding. BPB Publications, 2021. 470 p.

21. Uzayr S. b. Code Optimization. *Frontend Development*. Boca Raton, 2022. P. 127–132. URL: https://doi.org/10.1201/9781003309062-6 (date of access: 23.05.2025).