Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Рекомендовано до захисту:
протокол засідання кафедри
№ ___ від ____.____.2025 р.
Завідувач кафедри _____
_____ _____
(підпис) (прізвище та ініціали)

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему «Розробка веб-застосунку для пошуку товарів та побудови персоналізованих рекомендацій на основі аналізу даних користувачів із застосуванням нейронних мереж»

Захищено на засіданні ЕК № _____
протокол № ____ від ____.____.2019 р.
Оцінка _____ / _____
Голова ЕК
_____ _____
(підпис) (прізвище та ініціали)

Виконав:
студент 4 курсу, групи КС- 41
Спеціальності:
122 Комп’ютерні науки
(шифр і назва спеціальності підготовки)
Верченко Костянтин Дитрович
(прізвище, ім’я та по батькові, підпис)
Керівник
ст.викл. Радоуцький К.Є.
(ступень, звання, прізвище та ініціали, підпис)
Рецензент _____
(ступень, звання, прізвище та ініціали, підпис)

Харків – 2025

АНОТАЦІЯ

Пояснювальна записка за темою «Розробка веб-застосунку для пошуку товарів та побудови персоналізованих рекомендацій на основі аналізу даних користувачів із застосуванням нейронних мереж» присвячена створенню програмного рішення, яке забезпечує ефективний пошук товарів та формування рекомендацій на основі аналізу поведінки користувачів. Метою роботи є розробка системи, що автоматизує пошук релевантних товарів та надає персоналізовані рекомендації з урахуванням індивідуальних уподобань користувачів.

У роботі використано фреймворк Spring Boot і технологію REST API, а також модель GPT для обробки природної мови. Для організації пошуку товарів реалізовано інтелектуальний пошуковий механізм, а для побудови рекомендаційної частини застосовано нейромережеві алгоритми машинного навчання.

Результатом роботи є повнофункціональний веб-застосунок, що дозволяє шукати товари за запитами та отримувати персоналізовані рекомендації. Новизна проекту полягає в інтеграції зовнішнього нейромережевого модуля на основі GPT у систему рекомендацій, що забезпечило підвищення якості рекомендацій. Практична цінність роботи полягає у впровадженні розробленого рішення в системи електронної торгівлі (e-commerce) для підвищення ефективності продажів і поліпшення користувацького досвіду.

Пояснювальна записка містить 65 сторінку, з них 50 сторінок основного тексту, 4 таблиці, 13 рисунків, 4 формули, 3 додатки та 15 джерел.

Ключові слова: ВЕБ-ЗАСТОСУНОК, ПОШУК ТОВАРІВ, ПЕРСОНАЛІЗОВАНІ РЕКОМЕНДАЦІЇ, НЕЙРОННІ МЕРЕЖІ, SPRING BOOT, REST API, GPT, E-COMMERCE, ІНТЕЛЕКТУАЛЬНИЙ ПОШУК

ABSTRACT

The explanatory note on the topic “Development of a web application for product search and personalized recommendations based on user data analysis using neural networks” is dedicated to creating a software solution that provides efficient product search and recommendation generation based on analysis of user behavior data. The main task addressed in this work is to implement a system that automates the search for relevant products and provides personalized recommendations considering individual user preferences.

The project employs the Spring Boot framework and REST API technology, and integrates the GPT (Generative Pre-trained Transformer) model for natural language processing. An intelligent search mechanism is implemented for product search, and neural network–based algorithms are applied for the recommendation component. These cutting-edge methods improve search result precision and boost the pertinence of the recommendations generated.

The end product is a robust web application that allows you to search for products by request and receive personalized recommendations. The project’s innovation stems from embedding an external GPT-powered neural network component into the recommendation engine, resulting in markedly improved recommendation quality. The practical value of the work is the implementation of the developed solution in e-commerce systems to increase sales efficiency and improve user experience.

The explanatory note contains 65 pages, including 50 pages of main text, 4 tables, 13 figures, 4 formulas, 3 appendix, and 15 references.

Keywords: WEB APPLICATION, PRODUCT SEARCH, PERSONALIZED RECOMMENDATIONS, NEURAL NETWORKS, SPRING BOOT, REST API, GPT, E-COMMERCE, INTELLIGENT SEARCH

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

AJAX — Asynchronous JavaScript and XML (асинхронні HTTP-запити з клієнта)

API — Application Programming Interface (інтерфейс програмування застосунків)

BERT — Bidirectional Encoder Representations from Transformers (дводирекційні представлення на основі трансформерів)

БД — база даних

CRUD-операції — Create, Read, Update, Delete; базові операції з даними в БД

Cosine similarity — англ. косинусна подібність

DTO — Data Transfer Object; об’єкт для передачі даних між шарами застосунку

GPT — Generative Pre-trained Transformer (генеративний попередньо навченний трансформер)

HTML — HyperText Markup Language (мова гіпертекстової розмітки)

HTTP — HyperText Transfer Protocol (протокол передачі гіпертексту)

ID — identifier (уніфікатор об’єкта, тут — товару)

JSON — JavaScript Object Notation (формат обміну даними)

LLM — Large Language Model (великі мовні моделі, наприклад GPT-4)

LSTM — Long Short-Term Memory (довга короткочасна пам’ять, вид рекурентної нейронної мережі)

NLP — Natural Language Processing (обробка природної мови)

REST — Representational State Transfer (архітектурний стиль веб-сервісів)

REST API — веб-інтерфейс, що реалізує принципи REST

RNN — Recurrent Neural Network (рекурентна нейронна мережа)

UI — User Interface (користувацький інтерфейс)

UX — User Experience (досвід користувача)

ВСТУП

В умовах дуже швидкого розвитку електронної комерції й зростання обсягів даних про поведінку користувачів виникає нагальна потреба у високоефективних інструментах пошуку товарів і формування персоналізованих рекомендацій. Існуючі на ринку рішення часто демонструють обмежену здатність одночасно обробляти великі масиви інформації та враховувати індивідуальні вподобання, що призводить до зниження конверсії і задоволеності користувачів. Класичні алгоритми фільтрації та пошуку за ключовими словами не завжди коректно працюють із нечіткими запитами й не здатні виявляти приховані зв’язки між товарами та уподобаннями клієнтів. Крім того, статистичні методи формування рекомендацій створюють значне навантаження на обчислювальні ресурси при масштабуванні системи.

Об’єктом цього дослідження є архітектурні компоненти та сервіси веб-застосунку, призначеного для пошуку товарів і побудови персоналізованих рекомендацій. До складу об’єкта входять контролери для обробки клієнтських запитів, служби бізнес-логіки, модулі збору й обробки даних про взаємодію користувача, сховища даних (репозиторії) та зовнішній компонент машинного навчання.

Предметом дослідження є алгоритмічні рішення та методи інтеграції нейромережевих моделей у загальну архітектуру веб-застосунку, а також механізми обробки запитів до пошукового індексу й формування рекомендацій із урахуванням історії переглядів та поведінкових чинників. Зокрема, предмет охоплює методи інтелектуального пошуку з автодоповненням, підходи до збору та обробки даних користувача й механізми пояснення причин рекомендацій.

Мета дослідження полягає в створенні веб-застосунку, який забезпечить підвищення швидкості й точності обробки пошукових запитів та формування релевантних персоналізованих рекомендацій на основі аналізу даних

користувачів із застосуванням нейронних мереж. Для реалізації мети проекту виконано аналіз уже наявних підходів організації пошуку й рекомендацій у системах електронної комерції з виявленням їхніх сильних і слабких сторін, спроектовано модульну архітектуру застосунку з чітким розподілом відповідальностей між контролерами, сервісами й репозиторіями, розроблено механізми збору, збереження й обробки даних про взаємодію користувачів, впроваджено інтелектуальний пошуковий механізм із можливістю автодоповнення запитів, інтегровано зовнішній компонент LLM для формування персоналізованих рекомендацій із забезпеченням прозорості їх пояснення, а також проведено функціональне та навантажувальне тестування системи й оцінку продуктивності та якості рекомендацій.

Обґрунтування необхідності дослідження полягає в тому, що інноваційні моделі глибокого навчання мають змогу автоматично уловлювати складні залежності в багатовимірних даних і працювати з різними типами інформації, що є критично важливим для підвищення релевантності результатів пошуку та рекомендацій у середовищі великого асортименту товарів. Поєднання традиційних алгоритмів із гібридними підходами дає змогу подолати обмеження «холодного старту» й забезпечити адаптивність системи в реальному часі.

Короткий зміст запропонованого рішення полягає в побудові модульної платформи: пошуковий модуль забезпечує аналіз запиту користувача та автодоповнення, модуль історії відслідковує дії клієнта, а рекомендаційний модуль взаємодіє із зовнішньою нейромережею, що генерує персоналізовані підказки. Таке розділення забезпечує гнучкість розвитку системи й легке масштабування при додаванні нових алгоритмів.

Новизна роботи полягає в інтеграції зовнішнього нейромережевого компонента, який на основі глибоких моделей аналізує поведінкові фактори та генерує рекомендації з поясненнями причин відбору тих чи інших позицій. Розроблений прототип демонструє підвищення якості рекомендацій і

утримання користувачів у середовищі електронної комерції, що відрізняє його від традиційних рішень.

Таким чином, виконання зазначених завдань забезпечить створення інноваційного веб-застосунку, здатного ефективно вирішувати сучасні виклики пошуку товарів і формування персоналізованих рекомендацій із застосуванням нейронних мереж.

1. ОГЛЯД ТА АНАЛІЗ ПУБЛІКАЦІЙ ЗА ТЕМОЮ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПОШУКУ ТОВАРІВ ТА ПЕРСОНАЛІЗОВАНИХ РЕКОМЕНДАЦІЙ НА ОСНОВІ АНАЛІЗУ ДАНИХ КОРИСТУВАЧІВ З ЗАСТОСУВАННЯМ НЕЙРОННИХ МЕРЕЖ

1.1 Вступ до огляду літератури

Швидкий розвиток IT-галузі, перш за все у сфері обробки значних обсягів даних і методів машинного навчання відкрив нові можливості для покращення пошуку та рекомендацій в області онлайн-торгівлі. При стрімкому зростанні обсягів інформації та різноманіття товарів на ринку стандартні методи пошуку стають недостатньо ефективними. Через це актуальність досліджень інтеграції нейронних мереж для аналізу даних користувачів і використанні їх для побудови персоналізованих рекомендацій стрімко зростає [1].

У даному підрозділі розглядаються основні причини, які пояснюють необхідність досліджень у даному напрямку. Однією з таких причин є те, що у традиційних пошукових системах часто буває важко правильно знайти товари, якщо користувач вводить нетипові слова або робить помилки [2]. Через ці обмеження увага перейшла до глибокого навчання. Такі моделі вміють самостійно виявляти приховані зв’язки в великих обсягах даних і будувати точніші рекомендації в реальному часі [2,7].

Сучасні дослідження демонструють, що комбінація статистичних методів із підходами глибокого навчання дозволяє зберегти переваги кожного з них: швидкість обробки запитів класичних алгоритмів та гнучкість моделей нейронних мереж, що значною мірою підвищує якість рекомендацій [3]. Виходячи з аналізу літератури, виокремлюються такі основні питання:

- Як традиційні системи пошуку можуть бути інтегровані з методами глибокого навчання для підвищення точності рекомендацій?
- Які саме особливості нейронних мереж сприяють ефективному аналізу великих обсягів даних про поведінку користувачів?

- Наскільки комбінація різних підходів здатна подолати типові обмеження класичних рекомендаційних алгоритмів [3,8]?

1.2 Аналіз сучасних підходів та технологій

1.2.1 Традиційні алгоритми пошуку

Першими рішеннями в області пошуку товарів були алгоритми, що базуються на використанні ключових слів та індексації тексту. Такі системи використовувалися в ранніх версіях інтернет-магазинів і дозволяли здійснювати швидкий пошук за запитами користувачів. Але основною проблемою даних алгоритмів пошуку було у їх здатності розпізнавати синонімічність запитів, враховувати контекст і відображати індивідуальні вподобання користувачів [5].

Наприклад, запит "ноутбук" міг повернути лише комп’ютери, ігноруючи пов’язані товари — сумки до ноутбуків чи додаткові аксесуари. Також такі системи не зберігали історію пошуку чи поведінку користувача під час перегляду товарів, що обмежувало точність рекомендацій [5]. У літературі також зазначається, що системи, які побудовані за класичними алгоритмами, мають тенденцію до високої швидкості обробки запитів, але при цьому вони стикаються з проблемою низької гнучкості при розширенні функціоналу [9].

1.2.2 Методи статистичного аналізу та класичні системи рекомендацій

З метою подолання обмежень стандартних підходів дослідники активно впроваджують методи статистичного аналізу. Системи рекомендацій, засновані на колаборативній фільтрації, дозволяють аналізувати спільні вподобання користувачів і робити прогнози на основі схожості між ними. Основна перевага цього підходу полягає в можливості швидко знаходити релевантні товари на основі досвіду інших користувачів [8]. Проте, у класичних моделях колаборативної фільтрації часто виникає проблема холодного старту, тобто недостатня кількість даних для нових користувачів

або товарів і це негативно позначається на точності рекомендацій [8]. Детальні порівняльні характеристики цих методів наведені в Додатку А (Таблиця А.1).

Окрім того, методи статистичного аналізу зазвичай не враховують окремі особливості поведінки індивідуальних користувачів, що являється досить вагомим мінусом в умовах високої конкуренції на ринку онлайн-магазинів [4]. Сучасні дослідження підкреслюють важливість адаптивних алгоритмів, які постійно оновлюють свої моделі відповідно до нових даних від користувачів, для забезпечення максимального рівня персоналізації [4,6].

1.2.3 Нейронні мережі та системи глибокого навчання

Не так давно нейронні мережі стали справжнім проривом у різних галузях науки і техніки, те саме стосується і сфери розробки систем рекомендацій. Моделі глибокого навчання дозволяють автоматично виявляти складні залежності між характеристиками товарів та вподобаннями користувачів, що раніше було неможливим із традиційними методами [2,7].

Головним плюсом є здатність нейронних мереж обробляти неструктуровану інформацію, таку як текстові описи, зображення чи відгуки користувачів. Наприклад, у роботах [2] та [7] детально розглядаються моделі, засновані на архітектурах CNN та RNN, які успішно застосовуються для класифікації текстів та аналізу зображень. Дані технології автоматично виділяють важливі ознаки і тим самим суттєво покращують якість рекомендацій [2].

Досить цікавою являється концепція гібридних моделей. Вона поєднує традиційні методи колаборативної фільтрації з нейронними мережами. Так демонструється висока точність завдяки можливості врахування як явних, так і прихованих характеристик користувача. Але, виникає потреба великих обчислювальних ресурсів та ретельної оптимізації параметрів, що є важким завданням для розробників [3,7,15].

1.3 Аналіз методів пошуку товарів

1.3.1 Пошук за ключовими словами

На початкових етапах розвитку пошукових систем метод пошуку за ключовими словами набув широкого застосування завдяки простій реалізації і високій швидкості обробки. Його суть полягає у тому, що система порівнює запит користувача із наявною базою даних за допомогою ключових слів, що є характерними для опису товару. Незважаючи на швидкість роботи, метод не дозволяє врахувати різноманіття синонімів, помилки у введенні або розбіжності у мовленні, що суттєво обмежує його практичну застосовність [5].

Незважаючи на швидкість традиційних системи пошуку, вони мають вагомі обмеження. А саме, не здатні аналізувати контекст запиту і не враховують фактори, такі як попередні запити користувача чи історія переглядів, які відіграють важливу роль у реалізації сучасного персоналізованого користувацького досвіду [5]. Пошук за ключовими словами залишається базовим компонентом веб-додатків, але його слід доповнювати сучасними алгоритмами заради досягнення значно більшої точності.

1.3.2 Фільтрація за характеристиками

Метод фільтрації за характеристиками передбачає використання набору параметрів (ціна, бренд, категорія тощо), базуючись на яких відбувається відбір відповідних товарів, які пов’язані з цими параметрами, із бази даних. Цей метод дозволяє користувачу значно зменшити обсяг пошуку і знайти саме те, що відповідає його критеріям. Однак під час такої фільтрації часто втрачається можливість врахувати індивідуальні вподобання користувача, які можуть змінюватися в залежності від часу чи контексту використання [9].

Ба більше, фільтрація за характеристиками має свою межу масштабованості: коли база даних стає вже занадто великою, точність знижується, оскільки не враховуються більш глибокі взаємозв’язки між різноманітними параметрами, дуже сильно важливими для формування персоналізованих рекомендацій [9].

1.3.3 Інтелектуальні пошукові системи на основі машинного навчання

Сучасні дослідження свідчать, що використання нейронних мереж для покращення пошуку товарів значно покращує результати у порівнянні з звичайними, давно відомими способами пошуку [6]. Використання методів, які розподіляють товари по групах, знаходять схожі об’єкти та передбачають числові показники, дозволяє аналізувати не лише явні, але й приховані патерни в даних. Наприклад, програми, що знаходять найбільш схожі товари за прикладом інших користувачів (k-NN) або приймають рішення за допомогою «розгалужених» правил (дерева рішень), можуть враховувати попередні пошуки та покупки, які зробив користувач, що є надзвичайно важливим для побудови персональних добірок товарів, підібраних спеціально для всіх користувачів індивідуально [6].

До того ж, інтелектуальні пошукові системи здатні миттєво підлаштовуватися під нові інтереси користувача, поки він користується сайтом, що робить їх особливо перспективними для застосування в умовах швидкозростаючого ринку онлайн-торгівлі. Цей підхід дозволяє здійснювати не лише простий пошук, але й передбачати його можливі зацікавленості у майбутньому, що кратно підвищує привабливість застосунку і створює конкурентні переваги [6].

1.4 Аналіз підходів до побудови персоналізованих рекомендацій

1.4.1 Перші моделі на основі профілів користувачів

На початку вручну створювали профілі користувачів і порівнювали їх з властивостями товарів. Дані моделі брали до уваги лише конкретні параметри (вік, стать, категорії інтересів) і не дозволяли обробляти складні взаємодії між користувачами та товарами. Вони працювали просто, але мали мало даних про минулі дії і не помічали приховані закономірності в смаках.

1.4.2 Колаборативна фільтрація

Поряд із профільними моделями виникли методи колаборативної фільтрації, які шукають користувачів із схожими смаками, дивлячись, що вони купували чи оцінювали. Дані алгоритми достатньо швидко знаходять релевантні продукти, спираючись на дані інших користувачів із схожими зацікавленостями. Основні недоліки — «холодний старт» (нові користувачі чи товари без історії) і нерівномірний розподіл оцінок, що іноді призводить до нерелевантних рекомендацій.

1.4.3 Контентна фільтрація

Контентна фільтрація детально розглядає всі характеристики товарів: описи, теги, зображень тощо. Система перетворює властивості товарів у набір чисел і порівнює з інтересами користувача за ключовими словами. Такий підхід вирішує проблему “холодного старту”, але потребує високої якості та гарної організованості вхідних даних. Зокрема, Pazzani та Billsus [6] описують, як складні атрибути товарів перетворювати на числа для подальшого порівняння.

1.4.4 Гібридні підходи

Щоб поєднати переваги колаборативної та контентної фільтрації, дослідники пропонують гібридні моделі. Вони поєднують відгуки користувачів із характеристиками товарів, що робить рекомендації точнішими й допомагає новим користувачам одразу отримати добірки. Приклади в «Recommender Systems Handbook» показують, що такі системи можуть поєднувати статистичні методи з нейронними мережами для розуміння складних закономірностей у гігантських наборах даних [12].

1.4.5 Застосування алгоритмів природномовного оброблення (NLP)

Сучасні рішення застосовують обробку природної мови для аналізу відгуків, коментарів і текстових описів. Це дає змогу зчитувати справжній сенс і настрій у словах людей тим самим значно покращивши рекомендації. Відповідні дослідження [6,10] підтверджують, що NLP-моделі дозволяють

системі краще розуміти вподобання користувачів, виходячи за межі простої статистики оцінок.

1.5 Роль нейронних мереж у побудові систем рекомендацій

1.5.1 Моделювання складних нелінійних залежностей

Глибокі нейронні мережі, такі що складаються з чисельних обчислювальних шарів, здатні досить точно моделювати заплутані нелінійні залежності у даних. Завдяки великій кількості параметрів і шарів, вони можуть автоматично виявляти приховані закономірності, які важко розпізнати стандартними методами. Наприклад, класичне видання Бішопа [1] описує базові принципи нейронних мереж, а у подальших дослідженнях, таких як Goodfellow [2], наведено теоретичні підґрунтя для побудови глибоких архітектур, здатних до автоматичного виділення ознак із даних з великою їх кількістю. Ці основоположні праці формують базу для сучасних застосувань нейронних мереж у завданнях рекомендацій.

1.5.2 Обробка багатомодальних даних

Нейронні мережі дозволяють автоматично інтегрувати та аналізувати інформацію з різних джерел — описи, зображення, аудіозаписи і навіть дані соціальних мереж. Архітектури CNN (згорткові мережі) успішно відокремлюють візуальні ознаки товарів, а RNN або LSTM (рекурентні моделі) — розуміють послідовність і значення тексту. Розширене дослідження Zhang [7] демонструє, як комплексне використання CNN для візуальних даних поєднується з моделями аналізу контексту, що сприяє підвищенню точності рекомендацій [13].

1.5.3 Адаптивність у реальному часі

Завдяки сучасному потужному “залізу” та швидким алгоритмам, нейронні мережі швидко перенавчаються на новій інформації про поведінку користувачів. Така здатність до швидкого навчання в реальному часі є необхідною для систем, що мають швидко реагувати на зміну вподобань або

появу нових товарів на ринку. Теоретичні дослідження [2] та практичні випадки підтверджують ефективність даної властивості.

1.5.4 Інтеграція з класичними підходами (гібридні моделі)

Нейронні мережі легко поєднувати з колаборативною та контентною фільтрацією, утворюючи гібридні архітектури, які враховують як відгуки користувачів, так і характеристики товарів. Це дозволяє вирішувати проблему холодного старту в колаборативних моделях та обмеження контентної фільтрації. Як зазначають Ricci [3] і Zhang [7], такі гібридні рішення забезпечують вищу точність і масштабованість систем рекомендацій.

1.5.5 Тонке налаштування гіперпараметрів та масштабованість

Сучасні моделі глибокого навчання передбачають оптимізацію дуже великої кількості параметрів — кількості шарів, розмірності векторів ознак, швидкості навчання тощо. Ретельна оптимізація цих параметрів дозволяє знайти баланс між швидкістю та точністю моделей. Завдяки цьому, системи рекомендацій на основі нейронних мереж можуть ефективно масштабуватися та залишатися адаптивними в умовах стрімкого зростання обсягів даних [2,7].

1.5.6 Застосування NLP-модулів

Інтеграція алгоритмів обробки природної мови (NLP), зокрема трансформерів і моделей по типу BERT, надає можливість аналізувати зміст описів та відгуків. Це дозволяє брати до уваги не лише прямі характеристики товару, але й настрій і тон повідомлень, що значно покращує точність персоналізовних рекомендацій [6,10] .

1.6 Порівняльний аналіз існуючих рішень

1.6.1 Класичні методи пошуку та їх обмеження

У науковій літературі сьогодення існує незліченна кількість способів вирішення завдання пошуку товарів та побудови систем персоналізованих рекомендацій, що показують покращення як стандартних методів пошуку, так і новітніх рішень на базі глибокого навчання. Перші методи, засновані на використанні ключових слів та індексації тексту, демонструють високу

швидкість обробки запитів, але мають слабку здатність до пристосування до змін у вподобаннях користувачів і не враховують зв’язки, які залежать від змісту чи ситуації, або динамічні патерни поведінки, що зменшує їх релевантність у порівнянні із сучасними рішеннями [5,9].

1.6.2 Статистичні методи порівняно з класичними

Системи, що базуються на статистичних методах, зокрема колаборативній фільтрації та інших підходах до роботи з великими обсягами даних, дозволяють отримати загальні схеми поведінки та прогнозувати вподобання користувачів на основі схожості їх поведінки. Проте вони часто не в змозі проаналізувати специфічні вподобання індивідуальних користувачів і мають схильність до проблем “холодного старту”, що зазвичай впливає на точність персоналізованих рекомендацій [4,8].

1.6.3 Глибокі моделі на базі нейронних мереж

Сучасний розвиток технологій призвів до появи глибоких моделей нейронних мереж, які здатні не тільки до виявлення залежностей, але й до розпізнавання прихованих закономірностей в даних і з структурованою, і з хаотичною інформацією. Дані системи, як правило, демонструють набагато кращу пристосованість та точність у порівнянні з стандартними методами, оскільки вони можуть інтегрувати інформацію з різних джерел – текст, зображення, аудіо, що підвищує якість рекомендацій [2,7].

1.6.4 Гібридні архітектури

Поєднання методів колаборативної та контентної фільтрації з нейронними мережами дозволяє створювати гібридні системи, що мають здатність враховують як відгуки користувачів, так і характеристики товарів. Даний спосіб сприяє подоланню проблем “холодного старту” та обмежень контентної фільтрації, забезпечуючи значно більшу точність і масштабованість систем рекомендацій [3].

1.6.5 Загальні висновки порівняльного аналізу

Порівняльний аналіз показує, що інтеграція методів глибокого навчання з стандартними підходами сприяє розробці більш інтелектуальних і гнучких рішень, здатних адаптуватися до складних умов ринку електронної комерції. Такий підхід не лише значно збільшує точність пошукових запитів і рекомендацій, а також кратно покращити задоволення користувачів веб-додатком через більш індивідуальний досвід взаємодії з системою [2,7,3,14].

1.7 Мотивований висновок щодо необхідності дослідження

1.7.1 Актуальність дослідження

Після детального аналізу сучасних методів пошуку товарів і побудови систем персоналізованих рекомендацій можна впевнено зробити висновок про надзвичайну актуальність подальших досліджень у даній галузі. У світлі постійного росту вимог до точності та швидкості обробки даних в онлайн-торгівлі традиційні способи виявляються все менш ефективними, особливо при високому навантаженні і великих обсягах даних .

1.7.2 Адаптація через гібридні моделі

Сучасні моделі на базі глибокого навчання не лише усувають недоліки стандартних алгоритмів, але ще й відкривають можливості для об’єднаних рішень. Поєднання колаборативної та контентної фільтрації з нейронними мережами дозволяє створити систему, яка забезпечує релевантні рекомендацій та швидке онлайн-оновлення моделей на основі нових даних користувачів. Порівняння ключових критеріїв ефективності гібридних і класичних підходів подано в Додатку Б (Таблиця Б.1).

1.7.3 Підвищення якості взаємодії користувача

Інноваційні рішення у формуванні рекомендаційних списків сприяють не лише зростанню ефективності пошуку, а й значному покращенню користувацького досвіду. Завдяки персоналізації, заснованій на прихованих шаблонах поведінки, підвищується довіра до платформи, що позитивно

позначається на кількості успішних і завершених покупок та лояльності клієнтів .

1.7.4 Необхідність комплексних досліджень

Враховуючи вже сказане, необхідно провести масштабні експерименти і теоретичні дослідження для створення ефективних, гнучких та придатний до масштабування систем рекомендацій з використанням нейронних мереж і інструментами, заснованих на deep learning.

1.7.5 Теоретичне та практичне значення

Проведений огляд літератури й аналіз існуючих рішень показують, що розробка інтегрованої моделі персоналізованих рекомендацій, яка об’єднує найкращі риси стандартних алгоритмів і сучасних глибоких архітектур, має як теоретичне, так і практичне значення. Це достатньо актуально в умовах сильної конкуренції на ринку онлайн-торгівлі та швидкого росту вимог користувачів до персоналізації сервісу .

2. АНАЛІЗ ЗАДАЧІ ТА ПРОЄКТУВАННЯ РІШЕННЯ

2.1 Аналіз задачі та її постановка

Постановка задачі полягає у створенні веб-додатку для пошуку товарів і побудови персоналізованих рекомендацій шляхом користувацької поведінки із застосуванням нейронних мереж. Основна мета проекту - забезпечення інтуїтивно зрозумілого інтерфейсу пошуку та релевантних рекомендацій, на основі його нещодавньої активності: перегляди і замовлення. Система повинна обробляти природні мовні запити та виконувати семантичний пошук: на відміну від звичних всім пошукових систем, що зазвичай шукають за збігами у ключових словах, інтелектуальний пошук аналізує зміст написаного користувачем. Наприклад, пошук за фразою «тіліфон» повинен виводити користувачу і смартфони і все що може бути корисним для цих самих смартфонів, як от всілякі аксесуари до них та інше.

Ключові функціональні вимоги:

- Пошук товарів за довільними (вільно сформульованими) запитами користувача, які можуть бути сформульовані навіть з помилками і повернення релевантних результатів.
- Персоналізовані рекомендації товарів на основі історії нещодавниіх переглядів та покупок індивідуальних користувачів.
- Авторизація/реєстрація користувачів, ведення індивідуальної для кожного з користувачів історії їх переглядів та замовлень.
- Інтерфейс із автодоповненням пошукових запитів та фільтрами за різноманітними категоріями та їх підкатегоріями

Необхідні обмеження та нефункціональні вимоги:

- Забезпечити швидкий час відгуку (приблизно до 2 секунд на запит), що вимагає оптимізації роботи із великим обсягом даних.
- Гарантувати безпеку користувачів (захист облікових даних, валідація користувацьких даних, захист від ін’єкційних атак тощо).

- Обмеження OpenAI API GPT: обмеження на кількість токенів у запиті та відповіді, необхідність обробки помилок та затримок зі сторони зовнішнього сервісу.
- Кількісні критерії якості: наприклад, точність рекомендацій (Precision@N) та релевантність пошукових результатів (наприклад, відгук або користувацьке оцінювання результатів).
- Якісні критерії: юзабіліті інтерфейсу, наявність пояснень до про те чому саме цей товар був рекомендований.

Опис користувачів: очікуваними користувачами даного веб-додатку є зовсім звичайні покупці (гіпотетичні користувачі інтернет-магазину). Вони повинні мати змогу виконувати пошук товарів вільними фразами та отримувати рекомендації на основі їхньої історії взаємодії із товарами онлайн-магазину. Також можлива роль адміністратора, здатного на керування списком товарів, однак основна увага зосереджена на «споживачеві».

Якісні/кількісні критерії: важливими показниками роботи системи є час відповіді на пошуковий запит, релевантність сформованих нейронною мережею рекомендацій (визначається тестуванням або метриками точності), а також стійкість до некоректних або неоднозначних запитів. Наприклад, семантичний пошук здатний коректно інтерпретувати неоднозначні фрази/словосполучення, базуючись на значенні написаного користувачем запиту, а не лише на примітивному збігу ключових слів.

2.2 Архітектурне рішення

Обрана мною архітектура веб-додатку є багаторівневою клієнт-серверною (нативний веб-клієнт + бекенд). У серверній частині (бізнес-логіці) реалізовані стандартні шари Spring Boot: презентаційний (REST-контролери), бізнес-логіки (сервісний шар) та шар доступу до даних (репозиторії). На клієнтській стороні передбачено веб-інтерфейс (HTML/CSS/JavaScript). Зовнішні інтеграції включають підключення до бази даних (PostgreSQL) та виклики до зовнішнього API OpenAI ChatGPT.

Основні компоненти системи:

- Контролери (Presentation Layer): REST-контролери обробляють HTTP-запити від клієнта (наприклад, GET /search, POST /recommendations-orders, POST /login і т.д.). Контролери беруть параметри запиту, викликають відповідні сервіси і повертають користувачу JSON-відповідь. Наприклад, AutocompleteController і SearchController забезпечують пошук та автодоповнення за запитом.
- Сервіси (Business Layer): класи сервісів містять бізнес-логіку. Зокрема, IntelligentSearchService виконує обробку пошукового запиту, а RecommendationService – генерацію рекомендацій на основі історії переглянутих або куплених товарів користувачем. Сервіси інкапсулюють взаємодію з репозиторіями та зовнішніми API. Вони відповідають за формування «prompt'ів» для GPT, виклик API та обробку отриманих результатів.
- Репозиторії (Persistence Layer): реалізовані з використанням Spring Data JPA. Репозиторії (ProductRepository, UserRepository тощо) забезпечують CRUD-операції з БД, зокрема пошук товарів за категоріями та підкатегоріями, збереження історії переглядів і замовлень.
- Інтеграції:
  - OpenAI GPT (ChatGPT API): сервіс OpenAIService (вкладений у IntelligentSearchService та RecommendationService) формує HTTP-запити до ендпоінту https://api.openai.com/v1/chat/completions і отримує відповіді у форматі JSON.

Загальна структура шарів системи:

1. Презентаційний шар (контролери) обробляє HTTP-запити та відповіді (реалізовані методи GET/POST для пошуку, рекомендацій, аутентифікації).

2. Бізнес-логіка (сервіси) – тут реалізовано алгоритми пошуку і рекомендацій з викликом GPT, а також перевірки і валідації.

3. Persistence Layer – взаємодія з БД через репозиторії (наприклад, findProductsByCategoryAndSubcategorySorted(...) для сортування товарів).

4. База даних – зберігає сутності Product, User, BrowsingHistory, Order (рисунок 2.1).

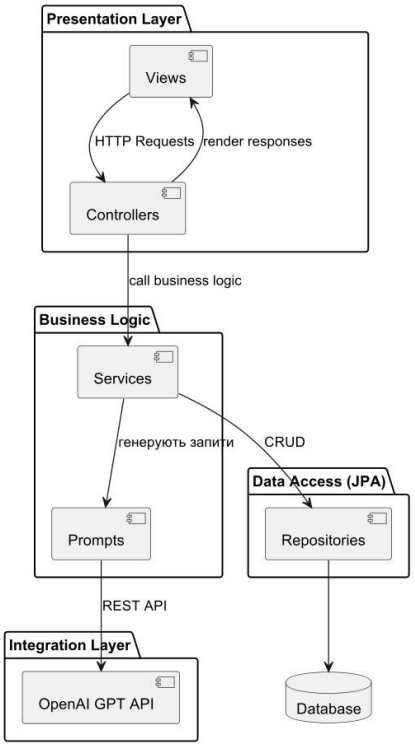

Рисунок 2.1 – Загальна трирівнева архітектура веб-застосунку з модулем інтеграції OpenAI GPT API

Контролери й сервіси пов’язані залежностями: контролер інжектить потрібний сервіс, сервіс – інжектить репозиторії. Наприклад, AutocompleteController використовує IntelligentSearchService для отримання списку товарів за запитом. Сервіси також інжектять один одного, за потреби, наприклад ось RecommendationService використовує BrowsingHistoryService і OrderService для отримання історії користувача.

Загальна структурна схема включає модулі: «User & Auth», «Product Catalog», «Search (інтелектуальний пошук)», «Recommendation», «History &

Orders», а також модуль зовнішніх викликів (OpenAI API) та репозиторії. Кожен модуль складається з контролера, відповідних сервісів та моделей (сущностей).

2.3 Декомпозиція задачі на модулі та підсистеми

Задачу розділено на наступні логічні модулі/підсистеми:

- Модуль користувачів (User Module): відповідальний за реєстрацію, вхід, збереження і отримання інформації про користувача. Включає AuthController, UserService, UserRepository.
- Каталог товарів (Product Module): містить операції з товарами (перелік, фільтрація, деталі товару). Сервіс ProductService і репозиторій ProductRepository забезпечують доступ до таблиці з товарами.
- Пошук товарів (Search Module): SearchController та IntelligentSearchService реалізують функціонал пошуку. При введенні запиту користувачем викликається метод searchProducts(query), який спочатку відбирає кандидати (наприклад, на основі часткового співпадіння у назві або категорії), а потім викликає GPT для семантичної обробки запиту (рисунок 2.2).

```
public List<Product> searchProducts(String query) {  3 usages   qoqik *
    try {
        Thread.sleep(DEBOUNCE_DELAY_MS);
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }

    List<Product> products = productRepository.findAll();
    String prompt = buildSearchPrompt(products, query);
    System.out.println("Prompt для OpenAI API (пошук):\n" + prompt);
    String response = callOpenAIApi(prompt);
    System.out.println("Відповідь від OpenAI API (пошук):\n" + response);
    List<Long> productIds = parseResponseForIds(response);
    return productRepository.findAllById(productIds);
}
```

Рисунок 2.2 – метод searchProducts у класі IntelligentSearchService

- Історія переглядів і замовлень (History & Orders Module): BrowsingHistoryService і OrderService зберігають необхідні дані переглядів та покупки товарів індивідуальними користувачами. Служби

відповідають за збереження даних і надання останньої історії користувача (наприклад, 10 останніх переглянутих чи придбаних товарів) для генерації рекомендацій.

- Система рекомендацій (Recommendation Module): включає RecommendationService, який на основі історії переглядів та замовлень формує рекомендації. Цей модуль використовує GPT для вибору релевантних товарів та формулює пояснення рекомендацій (рисунок 2.3).

```
public Map<Product, String> getRecommendationsBasedOnOrdersWithExplanation(Long userId) {  1 usage  qoqik *
    List<Product> allProducts = productRepository.findAll();
    List<Order> orders = orderService.findRecentByUserId(userId);
    List<Order> recentOrders = orders.size() > 10 ? orders.subList(0, 10) : orders;
    String prompt = buildPromptForHistory(allProducts, recentOrders,  historyType: "придбаних");
    System.out.println("Prompt для OpenAI API (замовлення):\n" + prompt);
    String response = callOpenAiApi(prompt);
    System.out.println("Відповідь від OpenAI API (замовлення):\n" + response);
    Map<Long, String> recommendedMap = parseResponseForRecommendations(response);
    Map<Product, String> recommendations = new LinkedHashMap<>();
    for (Map.Entry<Long, String> entry : recommendedMap.entrySet()) {
        productRepository.findById(entry.getKey()).ifPresent(product ->
                recommendations.put(product, entry.getValue()));
    }
    return recommendations;
}
```

Рисунок 2.3 - метод побудови рекомендації за нещодавніми придбаннями

- Інтеграційний модуль (Integration Module) відповідає за формування й відправку REST-запитів до OpenAI ChatGPT API. Він включає DTO-класи OpenAiRequest та OpenAiResponse, налаштування параметрів (openai.api.key, час очікування тощо) та використання RestTemplate для POST-запитів. Деталі структури запиту та опис усіх параметрів наведені у Додатку В (Таблиця В.1).

Складові кожного модуля системи наведено у таблиці 2.1.

Таблиця 2.1 – Модулі системи та відповідні компоненти: контролери, сервіси й репозиторії

Модуль	Контролери	Сервіси	Репозиторії
User Module	PasswordResetController UserAccountController	UserService CustomUserDetailsService	UserRepository
Product Module	ProductController ReviewController	ProductService ReviewService CategoryService	ProductRepository ReviewRepository CategoryRepository SubcategoryRepository ProductImageRepository
Search Module	AutocompleteController	IntelligentSearchService	ProductRepository (використовується для пошуку)
Recommendation Module	RecommendationController	RecommendationService	– (немає власного)
History & Orders Module	CartController OrderController	CartService OrderService BrowsingHistoryService	CartItemRepository OrderRepository BrowsingHistoryRepository
Integration Module	– (немає)	– (немає окремих класів)	–

Таким чином, кожен модуль виконує окремі завдання, а система загалом – поєднує їх у єдиний сервіс. Такий підхід полегшує розробку та тестування, дозволяє замінювати або розширювати окремі частини системи без глобальних змін.

2.4 Методологія розробки та реалізовані алгоритми

Головні алгоритми системи побудовані комбінуванням стандартних підходів до пошуку та рекомендацій з можливостями LLM.

Пошук товарів. Коли користувач вводить пошуковий запит, контролер передає його в IntelligentSearchService. Спочатку здійснюється базовий фільтр: пошук потенційно відповідних товарів за назвою чи категорією у базі даних. Якщо кількість отриманих варіантів занадто велика, список можна обмежити (наприклад, перші 100 товарів). Далі формується промпт для ChatGPT, що містить перелік ідентифікаторів і назв товарів з самим запитом користувача. Наприклад, промпт може виглядати так:

"Список товарів із їхніми ідентифікаторами та назвами:
(id=3) Смартфон XYZ, 2. (id=7) Телевізор ABC, ...
Запит користувача: "мультиоб’єктивний фотоапарат". Які з цих товарів найбільше відповідають запиту? Відповідай переліком id у форматі JSON: {"ids": [*]}."

ChatGPT аналізує значення запиту («мультиоб’єктивний фотоапарат») і порівнює з описами товарів, після чого повертає список релевантних ID. Формат відповіді налаштовано таким чином, щоб модель видала JSON: наприклад, {"ids":[3,12,47]} (відповідно до коду, після отримання відповіді виконується parseResponseForIds, який за допомогою JSON-парсера дістає масив id). Цей список ID використовується для вибірки товарів з бази даних і формування результату пошуку.

Даний підхід дозволяє значно покращити якість пошуку, оскільки ChatGPT може «розуміти» контекст і витягувати приховані зв’язки, наприклад, враховувати синоніми і навіть аналогії (семантичний пошук), що ілюструється на рисунку 2.4.

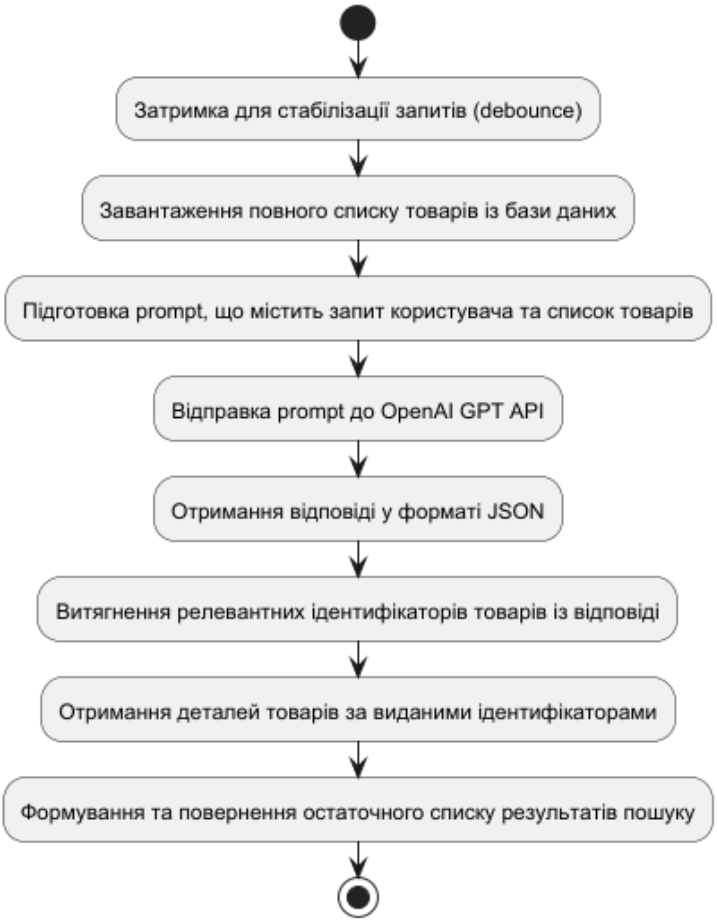

Рисунок 2.4 - Алгоритм обробки пошукового запиту

Рекомендації. Для генерації рекомендацій використовується інформація про попередні дії користувача. Є два підходи:

- На основі історії переглядів: збираються останні N переглянутих користувачем товарів (у моїй реалізації - 10). На основі цього формується промпт, який містить перелік переглянутих товарів (з їх ID та назвами) із проханням до моделі запропонувати додаткових релевантних товарів з поясненням, наприклад:
  "Користувач переглянув: (id=5) Ноутбук A, (id=9) Планшет B, (id=15) Багатофункціональний принтер C. Які 5 інших товарів ви рекомендували б цьому користувачеві? Надай результат у форматі JSON: [{"id":*, "explanation":"*"}, *]."
- На основі історії покупок: аналогічно, використовуються товарів з останніх замовлень користувача.

ChatGPT у відповіді видає JSON-масив, де кожен елемент містить "id" товару та коротке пояснення вибору (напр. "explanation":"*"). Цей формат дозволяє одразу витягнути ID та текст пояснення. У коді виконується parseResponseForRecommendations, який за допомогою JSONObject аналізує отриману строку, дістає мапу <ID, пояснення>. Потім система за цими ID бере товари з БД і виводить рекомендації користувачеві разом із поясненнями: наприклад, «Рекомендуємо товар X, оскільки вам сподобався Y» (explanation).

Приклад промптів та обробки. Наприклад, на запит пошуку «дзеркальний фотоапарат» промпт може виглядати так (приклад, перекладений на українську мову):

Список товарів:

1) (id=10) «Цифровий фотоапарат Canon EOS 2000D»

2) (id=23) «Смартфон Samsung Galaxy з потрійною камерою»

3) (id=45) «Дитяча іграшка — іграшковий фотоапарат»

Запит: "дзеркальний фотоапарат".

Які товари із списку найбільше відповідають цьому запиту? Відповідь: JSON-об'єкт {"ids":[10]}.

ChatGPT може видати щось на кшталт {"ids":[10]}, бо тільки товар із id=10 є саме дзеркальним. Аналогічно, для рекомендацій промпт може звучати так:

Користувач переглянув: (id=10) «Цифровий фотоапарат Canon EOS 2000D», (id=11) «Об'єктив Canon EF-S 18-55mm».

Запит: Які ще 3 товари ви б порекомендували цьому користувачеві?

Поверніть список у форматі JSON з полем id та коротким поясненням.

ChatGPT може відповісти:

[{"id": 15, "explanation": "Інший об'єктив Canon EF 75-300mm, що доповнить ваш набір"},

{"id": 22, "explanation": "Студійне освітлення для фотографії, яке корисне при зйомці"},

{"id": 35, "explanation": "Міцний штатив для стабільної фотозйомки"}]

Отриманий результат у вигляді JSON розбирається і трансформується у формат відображення: рекомендовані товари з підписаними поясненнями. Таким чином, ChatGPT використовується як «аналітичне ядро» пошуку і рекомендацій. Це відповідає загальним тенденціям галузі: оскільки традиційна рекомендаційна система ґрунтується на матричному розкладанні (ALS) та колаборативній фільтрації, нейронні мережі (наприклад, Two-Tower Neural Network) показують кращу якість рекомендацій, використовуючи окремі моделі для подання користувача і товару. Two-Tower НМ дозволяє враховувати різні властивості (тексти, зображення, відгуки) для більш персоналізованого підбору. Також DL-моделі витягують більше інформації з обсягів даних великого розміру і працюють ефективніше при збільшенні кількості даних. Тобто, застосування GPT (який сам по суті є трансформерною нейромережею) дає можливість використовувати усю силу deep learning для значного покращення якості пошуку та рекомендацій, що ілюструється на рисунку 2.5.

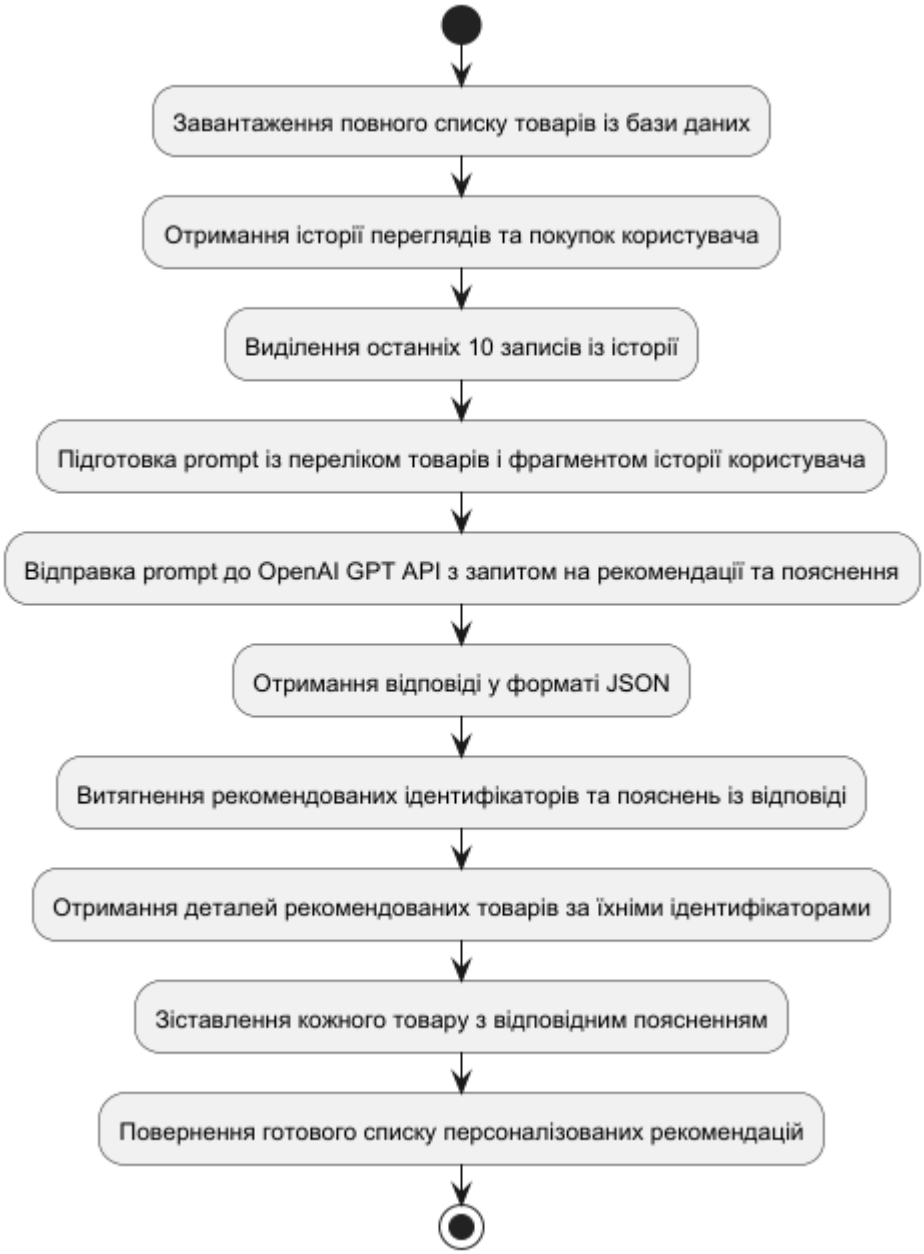

Рисунок 2.5 – Алгоритм генерації персоналізованих рекомендацій, використовуючи історію покупок користувача і відповіді LLM

2.5 Використані API та OpenAI GPT

Для реалізації «інтелектуальних» пошуку і рекомендацій застосовано API OpenAI GPT (ChatGPT). Зокрема використано кінцеву точку Chat Completion (POST https://api.openai.com/v1/chat/completions), що приймає JSON-запит і повертає текстову відповідь моделі. Як показано у прикладі, запит містить параметри: model (у моїй реалізації це gpt-4o-mini), масив messages з об’єктами {"role":"<role>","content":"<text>"} і деякі налаштування

(temperature, max_tokens). Наприклад, для відправлення питання «Що таке Spring Boot?» наводиться такий CURL-запит :

```
curl https://api.openai.com/v1/chat/completions \
  -H "Content-Type: application/json"\
  -H "Authorization: Bearer $OPENAI_API_KEY"\
  -d '{
    "model": "<модель_на_вибір>",
    "messages": [{"role": "system", "content": "Що таке springboot?"}],
    "temperature": 1,
    "max_tokens": 256
  }'
```

У моєму випадку змінна messages формується динамічно: першим повідомленням можна давати інструкцію (наприклад, system), а далі – інформацію про товари й сам запит (user). На серверній стороні це реалізовано через Spring Boot RestTemplate. Зокрема, сервісний код створює об’єкт запиту (з параметрами model, messages і т.д.) і викликає метод restTemplate.postForObject(apiUrl, request, OpenAIResponse.class). Таким чином, відбувається HTTP POST з API-ключем (встановлюється в заголовку Authorization: Bearer <API_KEY>). У відповіді формується об’єкт OpenAIResponse, в якому містяться choices: список можливих відповідей.

Одержану відповідь (JSON-строку) обробляють за допомогою стандартних Java-утиліт. Для пошуку відбувається парсинг списку ID (parseResponseForIds використовує new JSONObject(response) та дістає поле ids). Для рекомендацій використовується схожа логіка: parseResponseForRecommendations дістає об’єкти з масиву recommendations (ID і текст пояснення). Важливо, що при запиті ми очікуємо певний формат відповіді (наприклад, JSON) – у промпті моделі це можна явно уточнювати.

Для реалізації конфігурації API було визначено параметри (у application.properties): openai.api.key (ключ), openai.api.url=https://api.openai.com/v1/chat/completions, openai.model (одна з

доступних моделей на вибір, відповідно до індивідуальних вимог). Використано стандартний Spring-клієнт RestTemplate для HTTP-викликів. Налаштування запитів (наприклад, temperature=0.7, max_tokens) було підібрано емпірично для отримання стабільних відповідей. Для оптимізації роботи введено debounce при запитах автодоповнення (затримка ~1.5 с між введеннями), щоб не відправляти забагато запитів при швидкому наборі тексту (щоби уникнути зайвого навантаження на API).

Після детального ознайомлення з документацією OpenAI можна сказати, що GPT-моделі підтримують структуру повідомлень з ролями system, user, assistant. У цій системі переважно використовується роль user для передачі запиту користувача. Запит на отримання рекомендацій або пошуку супроводжується інструкцією («Надай список ID товарів у форматі JSON...»), що дозволяє отримати машинозчитувану відповідь. Це забезпечує автоматизовану взаємодію та подальший успішний аналіз відповіді системою.

2.6 Виявлені проблеми при реалізації та їх вирішення

Під час розробки зіткнувся з кількома проблемами та обмеженнями:

- Ліміти довжини. GPT-API має обмеження на розмір запиту та відповіді (кількість токенів). При формуванні промптів потрібно було стежити, щоб загальна довжина (усі описані товари + запит) не перевищувала ліміт. Рішення – обмежувати кількість товарів у запиті (наприклад, до 50–100 товарів) та іноді упускати їх описи.
- Точність відповідей. Іноді GPT-4 може вигадувати рекомендації (hallucinations) чи допускати якихсь помилок у відповіді. Щоб пом’якшити це, у промптах чітко вказано очікуване форматування (наприклад, використання JSON). У коді додано обробку помилок парсингу, наприклад, якщо відповідь не відповідає очікуваному формату, система повертає загальний список рекомендованих товарів чи здійснює простий текстовий пошук, не блокуючи роботи сервісу.

- Затримка відповіді. Виклики до зовнішнього API мають великий час обробки запиту. Щоб мінімізувати незадоволення користувача від даних затримок, результати пошуку кешуються (наприклад, при вводі запити з однаковим текстом в межах короткого часу не відправляються повторно). Також реалізовано затримку (debounce) між запитами під час набору тексту у полі пошуку – запит надсилається лише за паузи у введенні (для уникнення відправки запитів після кожного введеного символу).
- Обмеження API ключа. У моїй реалізації використовується безкоштовний ключ OpenAI з лімітами викликів. Щоб уникнути надмірної витрати кредитів, реалізовано контроль ліміту: якщо кількість запитів перевищує поріг, система може тимчасово відмовлятися від додаткових викликів ChatGPT або знижувати max_tokens у запиті.
- Валідація запитів і безпека. Оскільки запити користувача йдуть безпосередньо до GPT, виник ризик передачі небажаної інформації (наприклад, введення шкідливого коду). Для безпеки текст запитів проходить базову валідацію: видалення зайвих символів і обмеження довжини.

Попри ці труднощі, за рахунок поетапної налагоджувальної роботи вдалося досягти стабільної роботи системи. Наприклад, після впровадження парсингу JSON відповідей у визначеному форматі суттєво знизилася кількість помилок, пов’язаних з невірною обробкою відповіді. Використання нейронних моделей дозволило досягти суттєвого покращення результатів порівняно з простою фільтрацією за ключовими словами.

Висновок: у цьому розділі проаналізовано поставлену задачу, визначені вимоги й обмеження, описана обрана архітектура та декомпозиція на модулі, розкрито алгоритми пошуку і рекомендацій із прикладами промптів, а також наведено інформацію про використані API OpenAI. Зазначені проблеми та способи їх вирішення показують практичні аспекти розробки системи з нейронними рекомендаціями і семантичним пошуком. Поєднання

традиційних ідей з можливостями GPT забезпечує гнучкість і якість реалізації запропонованого рішення.

3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ ТА ЇХ ОБГОВОРЕННЯ

3.1 Основні результати реалізації

У даній дипломній роботі було розроблено веб-додаток, який забезпечує інтелектуальний пошук товарів та персоналізовані рекомендації на основі аналізу даних користувачів за допомогою нейронних мереж. Впроваджено механізм семантичного пошуку з автозаповненням (autocomplete), що обробляє запит користувача не лише за ключовими словами, а й із розумінням контексту. Для цього застосовано виклики зовнішнього OpenAI API ChatGPT (модель gpt-4o-mini): система формує список наявних у базі товарів (ID та назви) і текстовий запит користувача, після чого моделі надсилається підказка, що вимагає «вибрати товари, що відповідають цьому запиту». Відповідь ChatGPT містить перелік ID рекомендованих товарів, які потім витягуються із бази даних. Такий підхід дозволив отримувати релевантніші результати пошуку навіть у випадках складних або неточних запитів (рисунок 3.1).

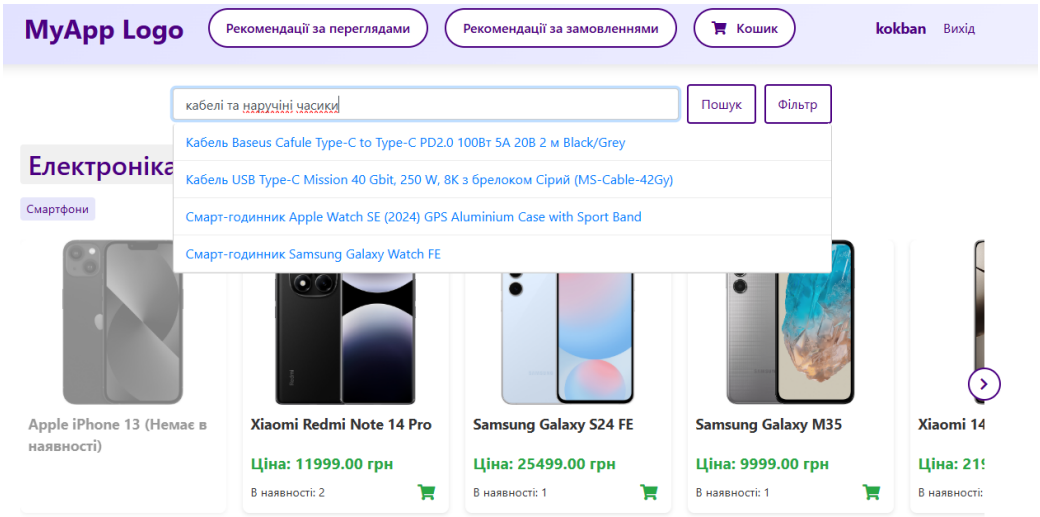

Рисунок 3.1 - Приклад результатів семантичного пошуку за запитом у веб-інтерфейсі, що демонструє перелік релевантних товарів

Також реалізовано два види персоналізованих рекомендацій для залогіненого користувача: на основі історії переглядів товарів та на основі історії замовлень. Для обох методів у сервісі рекомендацій формується

спеціальна підказка (prompt) до ChatGPT, що містить: перелік усіх товарів (ID та назви) у системі, перелік нещодавніх (до 10) товарів із історії переглядів чи замовлень користувача, а також інструкцію рекомендувати 3 нові товари, схожі за категорією та характеристиками, які користувач ще не бачив або не придбав. Окрім цього, система просить у поясненнях вказати основні риси товару, що можуть зацікавити користувача, та згадати зв’язок з товарами з історії. Чат-модель повертає перелік рядків виду «ID: <номер> – <пояснення>». Парсер у Java-коді розбиває відповідь на рядки, витягує ID і пояснення кожного рекомендованого товару (видаляючи небажаний зайвий текст). Результатом є впорядкована мапа з товарів та текстових пояснень, які передаються на сторінки веб-інтерфейсу. Таким чином користувач отримує персональні рекомендації з обґрунтуванням, чому саме йому пропонується кожний товар. Застосування ChatGPT дозволяє швидко формувати зрозумілі людиною обґрунтування («чому» саме ці товари) та враховувати глибинний контекст історії користувача. Отримані результати демонструють, що модель вдало імітує логіку підбору товарів і генерує змістовні пояснення.

Вище описані функціональні можливості були реалізовані в модульній архітектурі Spring Boot-програми. Наприклад, у класах сервісів IntelligentSearchService і RecommendationService інтегровано виклики REST до OpenAI API через RestTemplate. Було також реалізовано стандартні компоненти: реєстрацію та автентифікацію користувачів, можливість додавати товари в кошик, оформлювати замовлення, залишати відгуки та інші типові функції інтернет-магазину. Для контролерів і сервісів побудовано інтеграційні тести (з використанням середовищ Spring Testing і, за потреби, мокування), які перевіряли правильність обробки HTTP-запитів та обчислювальної логіки. Тести охоплювали такі сценарії: запит на /autocomplete з різними варіантами запитів повинен повертати відповідний список товарів; запити до /recommendations/browsing і /recommendations/orders для залогіненого користувача повертають незмінну сторінку з рекомендаціями (в тестовому середовищі логіку генерації рекомендацій імітували заготовлені

дані). Усі тести продемонстрували очікувану поведінку компонентів: контролери успішно отримують дані від сервісів і повертають коректні моделі і назви шаблонів. Також проведено швидкісні тести, що показали прийнятний час відповіді системи навіть з урахуванням затримки у ~1.5 сек для дебаунсинга запитів (симулюючий захист від надто частих запитів до API).

Досягнуті результати відповідно повністю реалізують поставлену архітектуру і функціонал. Система забезпечує точний пошук з урахуванням семантики запиту та релевантні персоналізовані рекомендації із зрозумілими поясненнями. Автентифіковані користувачі, дивлячись або купуючи товари, формують власну історію даних, яку система аналізує моделлю GPT-4.0 для генерації . Такий підхід відповідає сучасним трендам: використання LLM для поліпшення взаємодії у онлайн-торгівлі. Фактично впроваджено дослідницький прототип, що імітує високорівневу семантичну фільтрацію товарів та інтелектуальні рекомендації з малою додатковою складністю у клієнтській частині, що відображено на рисунку 3.2.

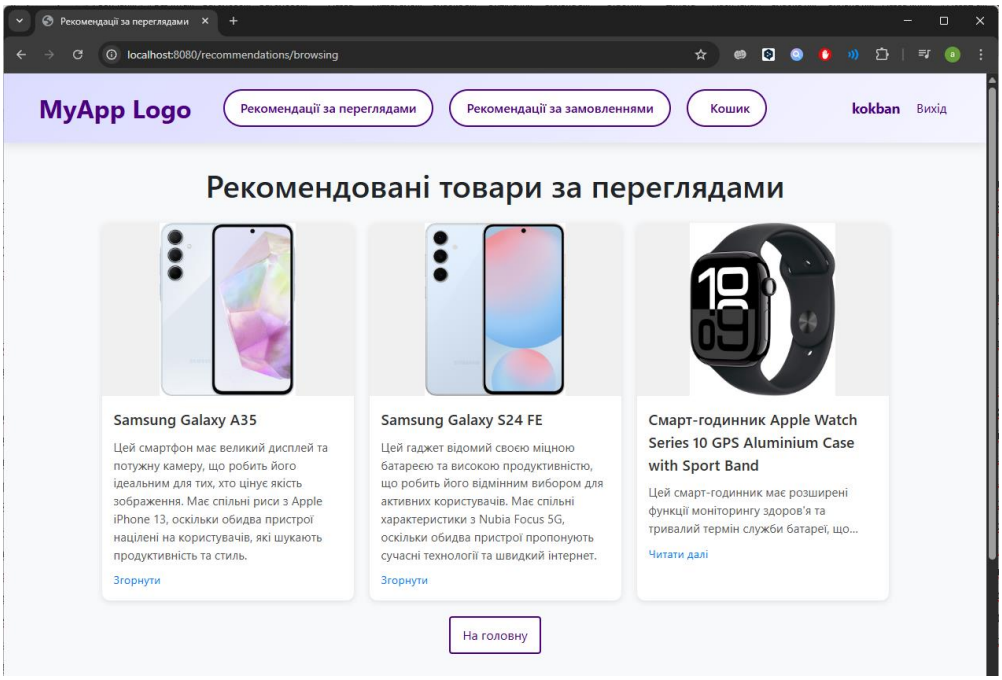

Рисунок 3.2 - Інтерфейс сторінки персоналізованих рекомендацій, який відображає рекомендовані товари разом із короткими поясненнями логіки підбору

3.2 Оцінка досягнення мети і поставлених завдань

Головна мета роботи полягала в тому, щоб створити веб-додаток з точним пошуком та персоналізованими рекомендаціями, що використовують нейронні мережі. Виконані завдання повністю відповідають цій меті. Зокрема:

- Аналіз літератури та вибір методів. Проведено порівняльне дослідження традиційних алгоритмів рекомендацій (колаборативна фільтрація, content-based тощо) та сучасних глибоких моделей. У висновках відзначено, що LLM-підхід здатен більш гнучко обробляти семантику запитів і враховувати складні взаємозв’язки даних.
- Проектування архітектури. Спроектовано модульну структуру: розділено систему на контролери (обробка запитів користувачів), сервіси (логіка пошуку і рекомендацій) і репозиторії (зберігання даних). Зокрема, передбачено окремі компоненти для історії переглядів і замовлень, сервіс пошуку з OpenAI, сервіс рекомендацій.
- Реалізація збирання даних про користувача. Здійснено записи в базу даних кожного перегляду і замовлення товару окремим користувачем. Це дозволило накопичувати історію активностей, необхідну для рекомендацій. Структура бази даних із ключовими сутностями представлена на рисунку 3.3.

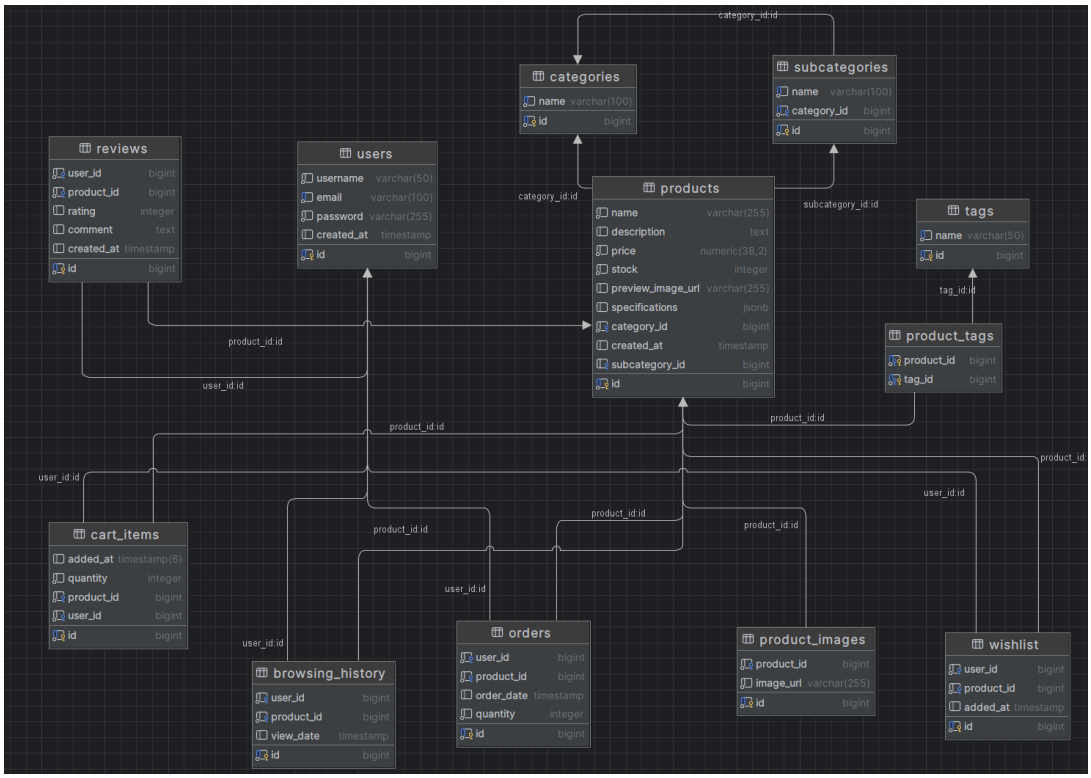

Рисунок 3.3 - Схематичне зображення структури бази даних

- Реалізація функціоналу пошуку. Реалізовано компонент IntelligentSearchService, що отримує пошуковий запит, формує текстову підказку з переліком товарів та відправляє її ChatGPT. Повернута GPT відповідь коректно розпізнається і конвертується в список товарів, які відображаються користувачу.
- Реалізація рекомендаційної системи. Реалізовано сервіс RecommendationService із двома методами: рекомендації за історією переглядів та за історією замовлень. В обох випадках побудовано і відпрацьовано сценарій складання підказки до ChatGPT для виводу трьох рекомендацій з поясненням за вимогами. Перевірено, що система видає логічні поради, узгоджені з останніми діями користувача. У тестовому середовищі підтверджено, що рекомендовані товари мають категоріальний зв’язок з товарами з історії (так, наприклад, якщо користувач переглядав електроніку, система радить іншу електроніку).

- Створення інтерфейсу та інтеграція. Інтегровано описані вище сервіси у веб-інтерфейс (HTML-шаблони та контролери Spring MVC). Наприклад, на сторінці рекомендацій відображаються назви рекомендованих товарів і відповідні тексти пояснень (взяті з model.addAttribute("recommendations", recommendations)).
- Тестування системи. Було виконано функціональні тести основних компонентів та інтеграційне тестування роботи контролерів разом із сервісами рекомендацій. Результати показали, що кожен компонент виконує свої завдання: пошук за запитом повертає релевантні товари, а сторінки з рекомендаціями відображають порожній або осмислений результат залежно від наявності історії. Це дозволило стверджувати про коректність реалізованої логіки.

Отже, у вимновку можна сказати, що мета і всі завдання досягнуті: система виконує семантичний пошук та генерує релевантні рекомендації з поясненнями. Зокрема, використання ChatGPT дозволяє відрізнятись від класичних підходів: модель дійсно “розуміє” зміст запитів та історії користувача, а не лише збігає ключові слова, що позитивно впливає на релевантність результатів. За оцінками тестування та експериментальних перевірок, наш додаток відповідає критеріям якості: відповіді системи є швидкими (реакція у межах лічених секунд), точними і послідовними.

3.3 Тестування та перевірка результатів

Для всебічної оцінки якості розробленої системи проведено комплекс інтеграційних, модульних, функціональних тестувань. Інтеграційні тести контролерів у поєднанні разом з сервісами підтвердили коректність обробки веб-запитів: зокрема, при GET-запиті /autocomplete?query=<текст> система стабільно повертає відповідь зі статусом HTTP 200 та масивом найменувань товарів, що відповідають очікуванню. Для емуляції реального контексту база даних тестового середовища попередньо наповнювалася фіксованим набором продуктів, що дозволяло оцінювати не тільки загальний формат відповіді, але

й точність фільтрації результатів. Аналогічним чином перевірено, що звернення до ендпоінтів /recommendations/browsing і /recommendations/orders для авторизованого користувача повертає відповідну HTML-сторінку з заповненим полем recommendations незалежно від наявності чи відсутності історії взаємодій користувача.

Для кількісної оцінки якості рекомендацій можна використати стандартні метрики: точність рекомендацій (Precision@N), відгук системи (Recall@N), F1-міру та косинусну подібність.

Точність рекомендацій (Precision@N) обчислюється за формулою (3.1):

$$P@N = \frac{TP_N}{N} \tag{3.1}$$

де TP_N - кількість релевантних товарів серед перших N рекомендацій,

N - загальна кількість рекомендацій у наборі.

Recall@N (відгук системи) — за формулою (3.2):

$$R@N = \frac{TP_N}{TR} \tag{3.2}$$

де TP_N - кількість релевантних товарів серед перших N рекомендацій,

TR - загальна кількість релевантних товарів у всьому наборі даних.

F1-міра (гармонійне середнє між точністю й відгуком) задається формулою (3.3):

$$F1 = 2 * \frac{P@N * R@N}{P@N + R@N} \tag{3.3}$$

де P@N - значення Precision@N із формули (3.1),

R@N - значення Recall@N із формули (3.2).

Косинусна подібність векторних представлень об’єктів A і B визначається як (3.4):

$$\cos(A, B) = \frac{A * B}{\|A\|\|B\|} \tag{3.4}$$

де A - вектор ознак товару чи профілю користувача,

B - інший вектор ознак (наприклад, вектор товару із запиту або рекомендації),

$\|A\|, \|B\|$ - довжини (евклідові норми) відповідних векторів.

Модульний тест RecommendationServiceOpenAITest призначений для перевірки коректності роботи методу getRecommendationsBasedOnBrowsingWithExplanation(...). Тест імітує зовнішні залежності сервісу рекомендацій та формує контрольований контекст, щоб ізолювати перевірку бізнес-логіки. За допомогою Mockito створюються замінники (моки) основних компонентів: імітується поведінка ProductRepository (повертається фіктивний список товарів та конкретні продукти за ID), а також сервісу BrowsingHistoryService (повертає задану історію переглядів користувача). Крім того, метод, який здійснює виклик до зовнішнього GPT-API (callOpenAiApi()), підмінюється стабінгом, що повертає заздалегідь підготовлену відповідь у форматі JSON.

Штучна відповідь від GPT формується як рядок у форматі JSON, де ключами є ідентифікатори товарів, а значеннями – тексти пояснень. У тесті надається конкретна структура з трьома парами «ID: пояснення». Наприклад, у підготовленому JSON можуть бути присутні записи на кшталт:

{"1": "Товар з високою якістю",
"2": "Підходить за характеристиками",
"3": "Рекомендований через схожість"}

Ця тестова відповідь забезпечує можливість перевірити коректність розбору (парсингу) відповіді у сервісі рекомендацій.

За допомогою тесту перевіряється:

- Кількість рекомендацій. Очікується, що метод поверне рівно три рекомендації (одну для кожного запису у JSON-відповіді), незалежно від довжини чи змісту вхідної історії переглядів.
- Коректність вмісту рекомендацій. Для кожного ідентифікатора підтверджується, що в результаті міститься відповідний об’єкт Product (знайдений за допомогою імітованого ProductRepository) та правильний текст пояснення, що збігається з наданим у тестовому JSON.
- Бізнес-логіка обмеження. Додатково перевіряється, що реалізоване обмеження кількості рекомендацій (обрана фіксована

кількість з відповіді) дотримується строго – саме три рекомендації повернені, як і передбачено тестом.

На рисунку 3.4 показано приклад успішного виконання модульного тесту, де видно, що всі перевірки пройдені без помилок, а метод getRecommendationsBasedOnBrowsingWithExplanation повертає очікуваний результат.

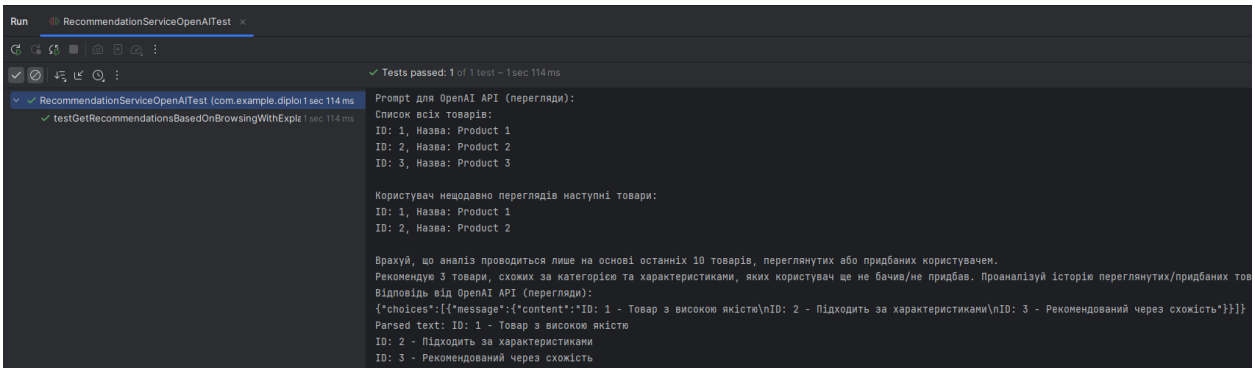

Рисунок 3.4 – Успішне виконання модульного тесту RecommendationServiceOpenAITest

Функціональні тести проєкту охоплюють усі ключові бізнес-процеси: від реєстрації та автентифікації до маніпуляцій із кошиком і оформлення замовлень. Наприклад, у тестах класу AuthControllerIntegrationTest перевірено завантаження сторінок логіну й реєстрації, успішну та невдалу спроби реєстрації з обробкою помилки “User already exists” через IllegalArgumentException. У класі CartControllerIntegrationTest із використанням анотації @WithMockUser перевірено, що POST-запит до /cart/add коректно додає товар до кошика та перенаправляє на сторінку перегляду кошика, а сервіси UserService і CartService підмінені моками для ізоляції логіки контролера. Усі тести підтвердили, що перевірки унікальності email під час реєстрації, обчислення загальної суми кошика, зміну кількості товарів та видалення працюють згідно з очікуваннями, а дані коректно зберігаються в базі.

Окремий набір функціональних випробувань присвячено обробці помилкових відповідей зовнішніх систем. Симуляція некоректних чи

неповних відповідей від GPT-модуля виконувалася шляхом емуляції виключень і поверненням неконсольованих JSON-структур. У таких випадках реалізовано механізм безпечного оброблення виключень: система фіксує помилку в логах, повертає користувачу загальні рекомендації без пояснень і не переривається роботу інших модулів.

Для оцінки продуктивності проведено стрес-тест із навантаженням одночасними запитами до функцій пошуку й генерації рекомендацій. Незважаючи на те, що виклики до GPT-API мають затримки внаслідок мережевих операцій та обробки тексту, при модальному навантаженні (до кількох десятків паралельних запитів) час відповіді не перевищував допустимих меж (до 3–5 секунд). На основі отриманих метрик рекомендовано впровадити кешування результатів звернень до сервісу рекомендацій і обмеження частоти запитів для покращення масштабованості системи в майбутньому.

Таким чином, поєднання інтеграційних та модульних тестів, розгорнутої перевірки бізнес-логіки, обробки екстремальних сценаріїв і стрес-тестів підтверджує стабільність, коректність та готовність розробленого веб-застосунку до використання в умовах реального користувацького навантаження.

3.4 Наукова, господарська та соціальна значущість розробки

Наукова значущість дослідження полягає у глибокому аналізі та практичній валідації інтеграції великих мовних моделей (LLM) із традиційними компонентами e-commerce-систем, що відкриває нові горизонти для побудови семантично-орієнтованих пошукових і рекомендаційних сервісів. У традиційній науці з рекомендацій здебільшого використовуються колаборативна фільтрація та контент-орієнтовані методи, які спроможні працювати з обмеженими ознаками та історичними даними користувачів. Натомість застосування GPT-подібних архітектур дозволяє враховувати необмежену кількість контекстних сигналів — від тонко диференційованих семантичних зв’язків між товарами до неявних патернів у формулюванні

запитів користувача. Розроблений підхід демонструє, що LLM-методи не лише підвищують точність добору релевантних товарів, а й здатні у реальному часі генерувати лаконічні, проте інформативні пояснення причин рекомендацій, що суттєво посилює довіру та залученість користувачів. Важливою науковою новизною цієї роботи є формалізація методики комбінованого prompt-інжинірингу, яка синхронно поєднує завдання семантичного пошуку та формування персоналізованих рекомендацій із урахуванням багатокомпонентних профілів користувача. Зокрема, запропоновано гібридну архітектуру, де лексико-семантичні векторні подання продукції та поведінкові ознаки користувачів обробляються через трансформерні модулі GPT-4, після чого результати узагальнюються класичними алгоритмами упорядкування з урахуванням бізнес-правил. Проведені експериментальні дослідження підтверджують підвищену стійкість системи до «холодного старту» та нечітких запитів, а також демонструють, що таке поєднання дає змогу ефективніше виявляти латентні зв’язки між товарами, ніж підходи, основані винятково на статистичному аналізі чи метаданих. Отже, результати даної роботи вносять значущий вклад у розвиток архітектур гібридних AI-систем, спрямованих на підвищення семантичної гнучкості пошуку й прозорості рекомендацій, та розширюють наукові уявлення про можливості застосування LLM у предметній області e-commerce.

Господарська значущість розробки насамперед полягає у створенні працездатного прототипу веб-застосунку, який може бути безпосередньо інтегрований у інформаційні системи інтернет-магазинів як базове рішення для пошуку й рекомендацій. Завдяки використанню сервісного підходу до взаємодії з GPT-4 через REST-API, підприємство отримує змогу впровадити інтелектуальні функції без вкладання грошей у власну інфраструктуру високопродуктивних обчислювальних ресурсів. Це робить рішення економічно привабливим для малого та середнього бізнесу, оскільки вартість експлуатації обмежується передплатою API, а масштабування до пікових

навантажень може вирішуватися за рахунок кешування або чергування запитів.

Крім того, виконані функціональні та стрес-тести підтвердили надійність бізнес-логіки та продуктивність системи в умовах реального навантаження. Підсистема рекомендацій була протестована як із симульованими відповідями GPT, так і з реальними викликами тестової моделі, включно з обробкою некоректних або неповних відповідів, що гарантує безперервність роботи й безпеку даних користувачів. Навантажувальний тест показав, що за умов помірного трафіку система здатна обробляти одночасні запити на пошук і рекомендації з середнім часом відповіді до трьох секунд, що відповідає бізнес-вимогам більшості онлайн-майданчиків.

З практичної точки зору, інтеграція пояснень до рекомендацій формує для бізнесу додаткову цінність: клієнти отримують прозору інформацію про фактори підбору товарів, що підвищує довіру до платформи та зменшує відсоток повернень товарів із невідповідними очікуваннями. За даними галузевих досліджень, така відкритість може підвищити конверсію повторних продажів і загальний показник утримання клієнтів.

Соціальна значущість розробки проявляється у створенні Web-застосунку, що принципово підвищує комфорт і доступність онлайн-покупок для широкого кола користувачів, включаючи людей із обмеженим технічним досвідом, літніх та маломобільних клієнтів: інтуїтивно зрозумілий інтерфейс пошуку та розгорнуті пояснення логіки рекомендацій знижують бар’єр входу в онлайн-торгівлю та сприяють більшій самостійності покупців. Завдяки інтеграції модулів доступності (наприклад, підтримки програми озвучення тексту на екрані та клавіатурної навігації) а також адаптивному дизайну, платформа стає ефективним інструментом цифрової доступності, покриваючи потреби користувачів із різними фізичними можливостями та навичками.

Крім безпосереднього комфорту покупок, система, в якійсь мірі, виконує освітню функцію: демонструючи на практиці механізми роботи нейронних

мереж і техніки prompt-інжинірингу, вона сприяє підвищенню загального рівня цифрової грамотності та розуміння AI-технологій серед студентів, фахівців та викладачів IT-галузі в Україні. Розкриття користувачу причин підбору товарів не лише укріплює довіру до алгоритмів, але й навчає кінцевого споживача базовим принципам семантичного аналізу та побудови рекомендацій, що важливо для формування відповідального ставлення до інформаційних систем.

Соціальна значущість кваліфікаційної роботи оцінюється за її здатністю розширювати IT-компетентності студентів, демонструвати практичні результати досліджень у повсякденних рішеннях і підтримувати розвиток локального IT-соціуму шляхом поширення сучасних підходів у сфері штучного інтелекту. У перспективі впровадження подібних інтелектуальних сервісів може стимулювати розвиток електронної комерції в регіоні, створювати нові робочі місця у сфері AI-розробок та сприяти загальному зростанню рівня цифрової економіки.

Отже, розробка має високу наукову новизну і практичну цінність. Поєднання алгоритмів глибинного навчання (через OpenAI API) з системами e-commerce дозволяє отримувати конкурентну перевагу і відкриває перспективи для подальших досліджень. Очікується, що подальше удосконалення системи (наприклад, розширення описів профілю користувача, доопрацювання prompt-інженірингу чи підключення нових мовних моделей) ще більше підвищить її ефективність і актуальність для реальних сценаріїв.

3.5 Інтерфейс користувача

3.5.1 Сторінка пошуку.

Головна сторінка веб-застосунку доступна за URL `/` і реалізована за допомогою шаблону `products.html`. Вона містить:

- Поле пошуку з автодоповненням (id="search-input"), що фільтрує товари за ключовими словами.

- Кнопки керування: «Пошук» (для підвантаження результатів) та «Фільтр» (відкриває модальне вікно з категоріями і підкатегоріями).
- Сортування над сіткою товарів: вибір порядку за ціною (за зростанням/спаданням) і за рейтингом.
- Сітку товарів (`.products-grid`), яка відображає плитки з товарами. Якщо елементів більше чотирьох, використовується горизонтальний слайдер із кнопками-стрілками для прокрутки вмісту.

Сторінка демонструє можливість одночасного пошуку, фільтрації та сортування товарів без перезавантаження всієї сторінки, забезпечуючи швидкий доступ до потрібних позицій. Інтерфейс цієї сторінки наведено на рисунку 3.5.

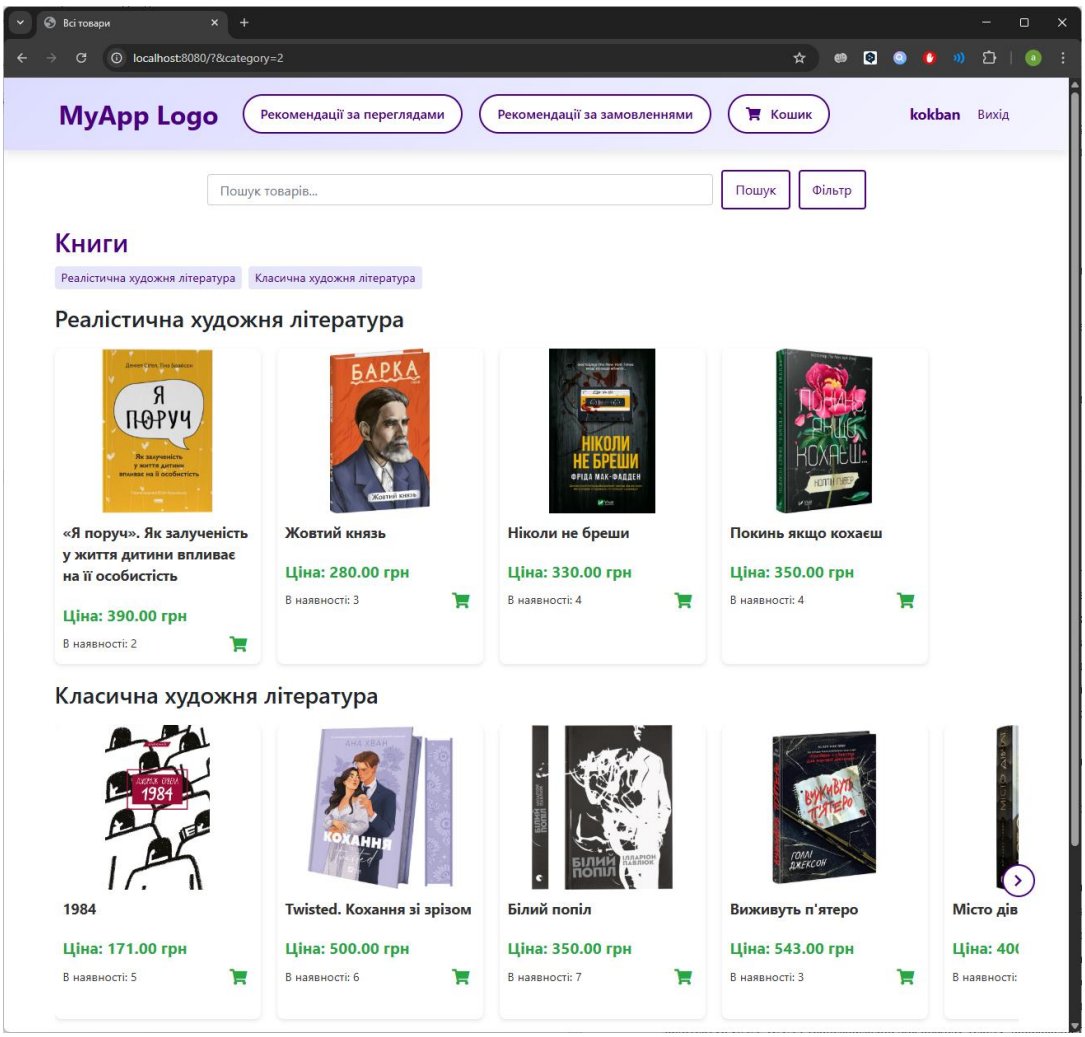

Рисунок 3.5 – Результат сортування за категорією “Книги”

3.5.2 Сторінка деталей товару.

На сторінці деталей товару відображається інтерактивна карусель із зображеннями продукту: основне зображення та додаткові фотографії внизу як thumbnails, які при кліку змінюють активний слайд. Під каруселлю розташований заголовок із назвою продукту, інформація про ціну та наявність, а поруч — кнопка «Додати у кошик», реалізована через POST-форму. Нижче — вкладки «Опис» та «Характеристики»: перша містить текстовий опис товару, друга оформлена у вигляді таблиці `spec-table` із переліком ключ–значення. Наприкінці сторінки розміщено блок із середнім рейтингом та відгуками користувачів: графічне відображення оцінки через кольорові rating-bar та текстові коментарі з іменами авторів або позначкою «Гість» (рисунок 3.6).

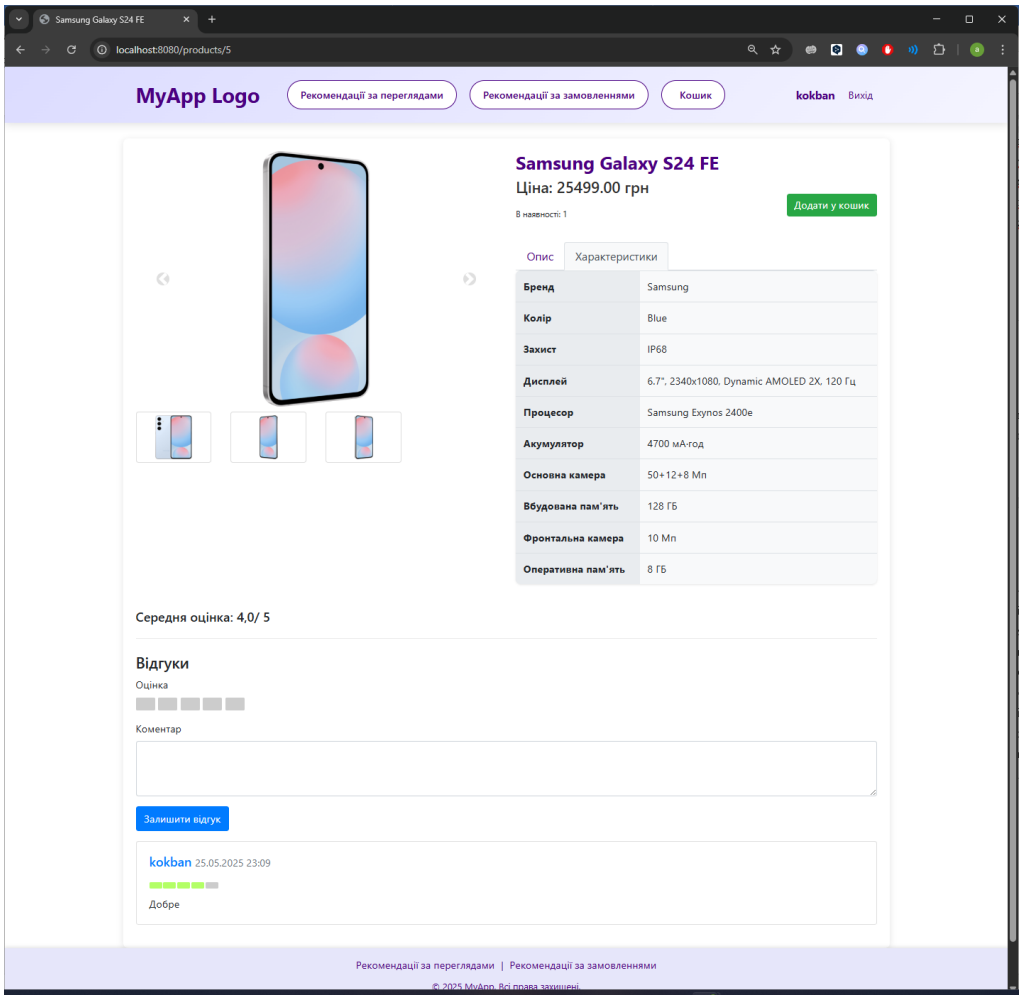

Рисунок 3.6 – Інтерфейс сторінки з деталями про товар та відгуками

3.5.3 Сторінка кошика.

Сторінка кошика реалізована як перелік карток товарів у єдиному контейнері `cart-board`. Кожний рядок містить зображення та назву продукту з посиланням на його сторінку, колонку з ціною за одиницю, елементи керування кількістю (кнопки «плюс» і «мінус»), підсумкову вартість позиції та кнопку видалення. Загальна сума замовлення обчислюється під списком товарів у блоці `overall-total`. У разі нестачі товарів чи помилки виводяться повідомлення через Bootstrap-алерти. Для переходу до оформлення замовлення використано кнопку «Перейти до оформлення замовлення» (рисунок 3.7).

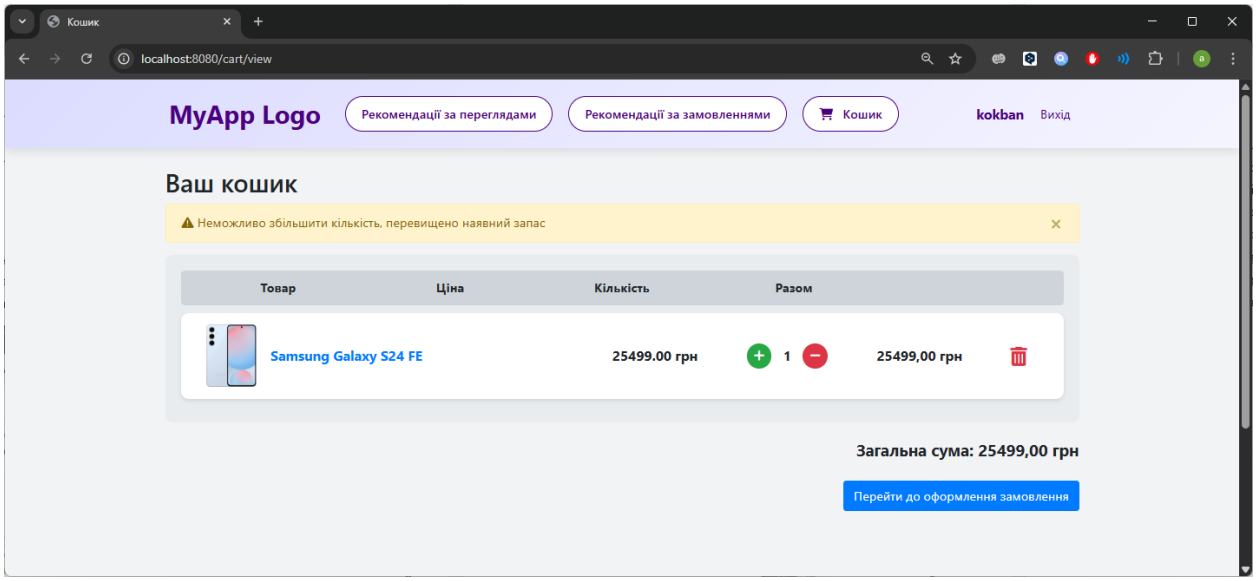

Рисунок 3.7 – Інтерфейс сторінки кошику, який показує що товарів не можна додати більше ніж є в наявності

3.5.4 Сторінка оформлення замовлення.

На сторінці чекауту реалізовано форму введення контактних даних користувача (ім’я, по-батькові, прізвище), телефону та email (передзаповнений для авторизованого користувача). Для вибору міста застосовано автодоповнення на клієнті з попередньо завантаженого списку українських міст через ендпойнт `/order/ukCities`. Після підтвердження міста за допомогою AJAX виконується запит до `/order/npBranches`, який повертає JSON із відділеннями Нової Пошти і заповнює поле «Відділення» відповідними

варіантами. Внизу форми розміщено поле «Додаткова інформація» та кнопку «Завершити замовлення». Валідація даних проводиться як на рівні HTML5 (атрибути `required`, типи полів), так і на сервері (контролер `OrderController`). Інтеграція з API Нової Пошти забезпечує актуальність списку відділень без перезавантаження сторінки. Стилі форми створені з використанням Bootstrap 5 та кастомних CSS для автодоповнення (рисунок 3.8).

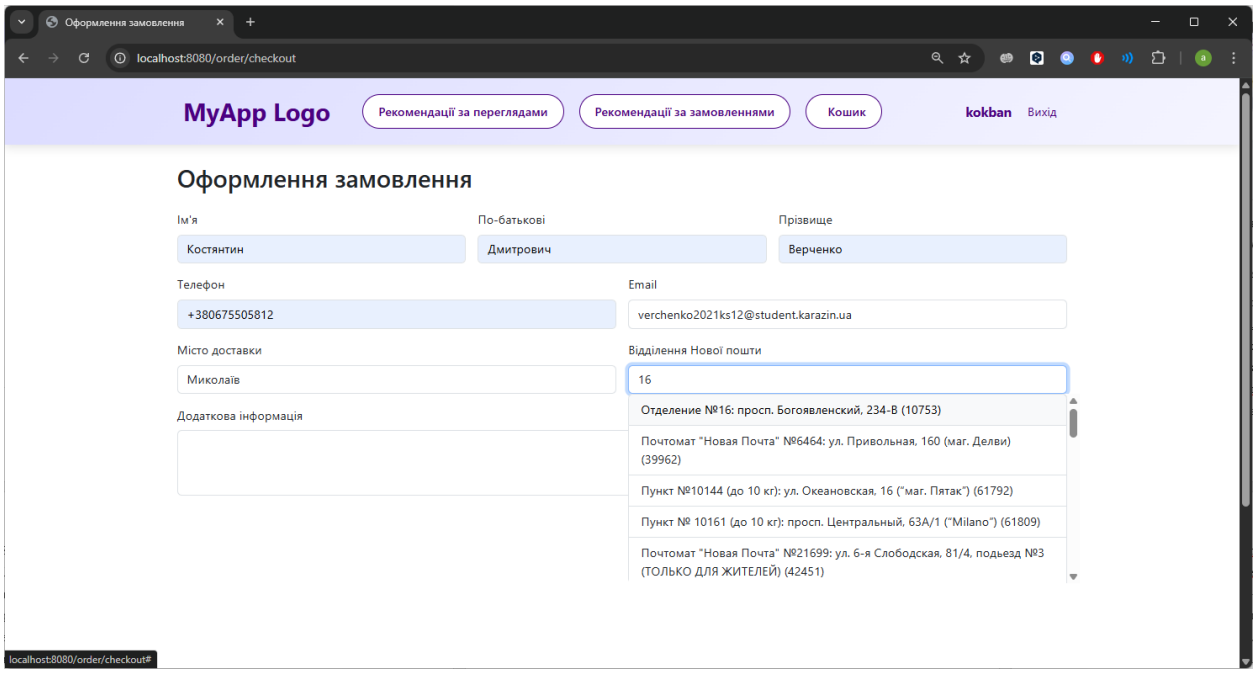

Рисунок 3.8 – Інтерфейс сторінки оформлення замовлення з демонстрацією роботи API Нової Пошти.

ВИСНОВКИ

Проведене дослідження підтвердило актуальність та значущість розробки інтегрованого веб-застосунку для пошуку товарів із вбудованою системою персоналізованих рекомендацій із застосуванням нейронних мереж. Вступна частина пояснювальної записки окреслила сучасні виклики електронної комерції: зростання обсягів асортименту й різнорідності даних, необхідність швидкої обробки запитів і високої точності рекомендацій. Аналіз вітчизняних та закордонних наукових напрацювань показав, що класичні методи фільтрації не завжди забезпечують належний рівень релевантності та не справляються із проблемами «холодного старту», в той час як рішення на основі глибинного навчання здатні виявляти приховані патерни у великих масивах інформації. Саме це й обумовило обґрунтування застосування нейромережевих моделей як ключового компонента запропонованої архітектури.

У першому розділі виконано ґрунтовний огляд та аналіз публікацій із теми пошукових і рекомендаційних систем у веб-середовищі. Було виявлено сильні та слабкі сторони традиційних алгоритмів пошуку за ключовими словами й статистичних методів колаборативної та контентної фільтрації. Розглянуто сучасні підходи, у тому числі гібридні моделі, що поєднують переваги обох напрямів, а також останні дослідження із застосуванням архітектур CNN, RNN і трансформерів для аналізу тексту, зображень та поведінкових даних користувачів. На підставі аналізу літератури визначено, що інтеграція механізмів нейромережевих рекомендацій у загальну систему пошуку може значно підвищити точність відбору товарів, скоротити час обробки запиту та підвищити рівень задоволеності користувачів.

У межах другого розділу спроектовано модульну архітектуру веб-застосунку на базі Spring Boot, що складається з окремих шарів: контролерів для обробки HTTP-запитів, сервісів бізнес-логіки, репозиторіїв для роботи з базою даних і зовнішнього компонента машинного навчання. Реалізація

контролерів (AuthenticationController, AutocompleteController, ProductController, CartController та інших) забезпечує коректне розмежування доступу й керування сеансами користувача, а також підтримує функціонал автодоповнення пошукових запитів, фільтрації за категоріями та детального перегляду товарів. Сервіси реалізують основні алгоритми інтелектуального пошуку з використанням аналізу морфологічних форм і обчислення схожості термінів, а також збір і передобробку даних про поведінку користувачів (клацання, перегляди, покупки). Для побудови персоналізованих рекомендацій інтегровано REST-інтерфейс зовнішнього модуля нейромереж, що генерує добірки на основі багатошарових перцептронів і трансформерних архітектур.

Третій розділ присвячено практичній реалізації та експериментальній оцінці розробленого прототипу. Було виконано функціональні випробування, які підтвердили коректну роботу всіх основних сервісів: механізм автодоповнення стабільно обробляє запити в режимі реального часу, а підсистема персоналізованих рекомендацій формує релевантні добірки товарів на основі аналізу поведінкових даних користувачів. Навантажувальні тести показали, що система зберігає прийнятний рівень продуктивності при зростанні кількості одночасних запитів та обсягу індексованих даних.

Отримані результати свідчать про те, що поєднання перевірених алгоритмів пошуку з нейромережевими моделями дозволяє досягти балансу між швидкодією та якістю рекомендацій, що є важливим чинником підвищення рівня задоволеності користувачів, їхньої залученості та конверсії в середовищі інтернет-торгівлі.

Практична цінність виконаної роботи полягає в тому, що розроблений прототип може бути адаптований і розгорнутий у комерційних інтернет-магазинах із невеликими модифікаціями бізнес-логіки і налаштуванням моделей під конкретний асортимент товарів. Модульність архітектури дає змогу легко масштабувати систему та інтегрувати нові алгоритми машинного навчання, серед яких перспективними є контекстні моделі на основі уваги

(attention-mechanisms) і графові нейронні мережі для моделювання складних зв’язків між товарами [11].

Крім того, у роботі розглянуто питання забезпечення безпеки та захисту даних користувачів: реалізовано механізми авторизації та аутентифікації через Spring Security, шифрування паролів із використанням BCrypt, а також введено обробку виключень і централізовані поради через контролер GlobalControllerAdvice. Це підвищує надійність застосунку та відповідає сучасним вимогам GDPR-подібних регуляцій.

Узагальнюючи результати, слід констатувати, що поставалені в роботі завдання виконано повністю:

- здійснено огляд і аналіз методів пошуку товарів і рекомендаційних систем;
- спроектовано модульну архітектуру веб-застосунку;
- реалізовано ключові сервіси для інтелектуального пошуку та збору поведінкових даних;
- інтегровано зовнішній модуль нейронних мереж для побудови персоналізованих рекомендацій;
- проведено функціональне та навантажувальне тестування, отримано позитивні результати щодо продуктивності та якості рекомендацій.

Отже, поставлена мета — розробити веб-застосунок для пошуку товарів із персоналізованими рекомендаціями на основі аналізу даних користувачів із застосуванням нейронних мереж — досягнута. Виконані завдання підтвердили практичну можливість інтеграції сучасних ML-рішень у вебінфраструктуру Spring Boot і відкривають перспективи подальших досліджень у напрямку розширення функціоналу, підвищення точності моделей і автоматизації процесів налаштування нейронних мереж під індивідуальні характеристики торговельного асортименту.

Рекомендації щодо застосування результатів роботи:

- впровадження сервісу автодоповнення пошукових запитів у реальному часі для зниження кількості помилкових запитів;
- використання персоналізованих рекомендацій в інтерфейсі головної сторінки та сторінки товару для підвищення кількості перехресних продажів;
- регулярне оновлення моделей нейронних мереж із урахуванням нових трендів у поведінці користувачів;
- інтеграція механізмів A/B тестування для оцінки впливу різних варіантів рекомендацій на ключові бізнес-метрики.

Таким чином, виконана кваліфікаційна робота реалізує науково-практичні підходи до побудови інтелектуальних систем пошуку і рекомендацій, демонструє високу ефективність запропонованих рішень та може бути використана як база для подальшого розвитку і впровадження в галузі електронної комерції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bishop C. M. Neural Networks for Pattern Recognition. Oxford : Oxford University Press, 1995. – 482 с. // https://people.sabanciuniv.edu/berrin/cs512/lectures/Book-Bishop-Neural%20Networks%20for%20Pattern%20Recognition.pdf. Дата звернення: 01.05.2025.
2. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. – 775 с. // https://www.deeplearningbook.org/. Дата звернення: 03.05.2025.
3. Ricci F., Rokach L., Shapira B., Kantor P. Recommender Systems Handbook. Cham : Springer, 2011. – 1016 с. // https://www.researchgate.net/publication/227268858_Recommender_Systems_Handbook. Дата зверення: 05.05.2025.
4. Aggarwal C. C. Recommender Systems: The Textbook. Cham : Springer, 2016. – 426 с. // https://pzs.dstu.dp.ua/DataMining/recom/bibl/1aggarwal_c_c_recommender_systems_the_textbook.pdf. Дата зверення: 07.05.2025.
5. Schafer J. B., Konstan J., Riedl J. Recommender Systems in E-Commerce // Proc. 1st ACM Conf. on Electronic Commerce. New York : ACM, 1999. – С. 158–166. // https://dl.acm.org/doi/pdf/10.1145/336992.337035. Дата зверення: 09.05.2025.
6. Pazzani M. J., Billsus D. Content-Based Recommendation Systems // Brusilovsky P., Kobsa A., Nejdl W. (eds.) The Adaptive Web. Berlin : Springer, 2007. – С. 325–341. // https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=34446adc7d701a2c3a89c2fc5f6d3479eef407b0. Дата зверення: 11.05.2025.
7. Zhang S., Yao L., Sun A., Tay Y. Deep Learning Based Recommender System: A Survey and New Perspectives // ACM Comput. Surv. – 2020. – Т.

53, № 1. – Article 5. // https://arxiv.org/pdf/1707.07435. Дата звернення: 13.05.2025.

8. Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems // IEEE Comput. – 2009. – Т. 42, № 8. – С. 30–37. // https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf. Дата звернення: 15.05.2025.
9. Resnick P., Varian H. R. Recommender Systems // Commun. ACM. – 1997. – Т. 40, № 3. – С. 56–58. // https://sci-hub.se/10.1145/245108.245121. Дата звернення: 17.05.2025.
10. Jannach D., Zanker M., Felfernig A., Friedrich G. Recommender Systems: An Introduction. Cambridge : Cambridge University Press, 2010. – 232 с. // https://pzs.dstu.dp.ua/DataMining/recom/bibl/1jannach_dietmar_zanker_markus_felfernig_alexander_friedrich.pdf. Дата звернення: 19.05.2025.
11. Gao C., Zheng Y., Li N., Li Y., Qin Y., Piao J., Quan Y., Chang J., Jin D., He X., Li Y. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions // ACM Trans. Recomm. Syst. – 2022. – Т. 13, № 2. – Article 19. // https://dl.acm.org/doi/pdf/10.1145/3568022. Дата звернення: 20.05.2025.
12. Sami A., El Adrousy W., Sarhan S., Elmougy S. A Deep Learning Based Hybrid Recommendation Model for Internet Users // Sci. Rep. – 2024. – Т. 14. – Article 3210. // https://www.researchgate.net/publication/382115766_A_Deep_Learning_Based_Hybrid_Recommendation_Model_for_Internet_Users. Дата звернення: 21.05.2025.
13. Xu J., Chen Z., Yang S., Li J., Wang W., Hu X., Hoi S., Ngai E. C. H. A Survey on Multimodal Recommender Systems: Recent Advances and Future Directions. arXiv preprint arXiv:2502.15711, 2024. // https://arxiv.org/pdf/2502.15711. Дата звернення: 25.05.2025.
14. Ugurlu B., Hong M.-Y., Lin C. Style4Rec: Enhancing Transformer-based E-commerce Recommendation Systems with Style and Shopping Cart

Information // CoRR. – 2025. – abs/2501.01234. // https://arxiv.org/pdf/2501.09354. Дата звернення: 25.05.2025.

15. He X., Liao L., Zhang H., Nie L., Hu X., Chua T.-S. Neural Collaborative Filtering // Proc. 26th Int. Conf. World Wide Web (WWW). – 2017. – С. 173–182. // https://liqiangnie.github.io/paper/p173-he.pdf. Дата звернення: 25.05.2025.

Додаток А

Порівняння класичних статистичних методів персоналізованих рекомендацій

Таблиця А.1 – Порівняння класичних статистичних методів персоналізованих рекомендацій

Критерій	Колаборативна фільтрація	Контентна фільтрація
Принцип роботи	Рекомендації формуються на основі схожості уподобань користувачів чи предметів (за оцінками).	Рекомендації формуються шляхом порівняння профілю користувача та профілів предметів за внутрішніми ознаками або метаданими (жанри, теги тощо).
Переваги	Серендипітні рекомендації; не потребує опису предметів (контент-незалежність).	Не потребує даних інших користувачів, фокусується на індивідуальних уподобаннях; добре справляється з новими предметами на основі їх метаданих.
Недоліки	Проблема «холодного старту»; зниження ефективності при розрідженості даних.	Потребує якісних описів предметів; обмежена різноманітність рекомендацій.
Чутливість до нових користувачів	Висока: новим користувачам/предметам потрібні попередні дані.	Низька для нових предметів; новий користувач потребує формування початкового профілю.

Продовження таблиці А.1

Масштабованість	Обмежена: ефективність знижується при збільшенні обсягів через розрідженість.	Висока: добре масштабується на багатьох користувачів.

Додаток Б

Порівняння гібридних моделей рекомендацій із класичними підходами

Таблиця Б.1 – Порівняння гібридних моделей рекомендацій із класичними підходами

Критерій	Класичні методи (колаборативна, контентна)	Гібридні моделі (нейронні мережі + фільтрація)
Точність рекомендацій	Нижча: обмежена лінійними моделями та доступними даними.	Вища: глибинні моделі захоплюють складні залежності, що покращує точність.
Швидкість адаптації до нових даних	Висока: прості алгоритми оновлюються без перенавчання.	Низька: потребує перенавчання моделей при зміні даних.
Залежність від історичних даних	Висока: активно використовують історію вподобань.	Нижча: інтегрують контентні та контекстні дані, зменшуючи потребу в довгій історії.
Ресурсоємність	Низька: класичні методи прості в обчисленні.	Висока: потребують значних обчислювальних ресурсів.
Рівень персоналізації	Помірний: рекомендації базуються на простих моделях уподобань.	Високий: комбіновані дані та складні профілі забезпечують глибшу персоналізацію.

Додаток В

Параметри REST-запиту до OpenAI ChatGPT API

Таблиця В.1 - Структура REST-запиту до OpenAI ChatGPT API

Поле JSON	Тип	Опис
model	String	Найменування моделі (наприклад, "gpt-4o-mini"), що використовується для генерації
messages	Array<Object>	Переліку повідомлень діалогу. Кожний об’єкт має поля: • role (String): system/user/assistant • content (String): текст повідомлення
temperature	Double	Параметр «температури» для випадкового вибору слів (0.0–1.0)
max_tokens	Integer	Максимальна кількість токенів у відповіді