Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інтернет-магазину комп'ютерної техніки з підсистемою рекомендації товарів

Виконав: студент IV курсу, групи СНс-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Пелішко О.І.
(підпис) (прізвище та ініціали)

Керівник _____ Палка О.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_8_» _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Пелішку Олегові Ігоровичу
(прізвище, ім'я, по батькові)
1. Тема роботи Розробка інтернет-магазину комп'ютерної техніки з підсистемою рекомендації товарів

Керівник роботи Палка Олег Вікторович, доктор філософії, асистент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*14*» *травня* 2026 року № *4/9-237*
2. Термін подання студентом завершеної роботи 28 червня 2026 р.
3. Вихідні дані до роботи Інтернет-магазин комп'ютерної техніки розроблений на базі PHP фреймворку Laravel. База даних MySQL. Рекомендаційна модель на базі алгоритму k-NN та косинусної схожості. Список використаних джерел.
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та методів реалізації рекомендаційних систем. 1.1 Аналіз електронної комерції у сфері продажу комп'ютерної техніки. 1.2 Класифікація рекомендаційних систем. 1.3 Колаборативна фільтрація. 1.4 Контентна фільтрація. 1.5 Алгоритм k-найближчих сусідів. 1.6 Гібридна модель рекомендацій. 1.7 Метрики оцінювання рекомендацій 1.8 Постановка задачі. 1.9 Висновки до першого розділу. 2. Проєктування програмної системи. 2.1 Загальна архітектура системи. 2.2 Проєктування бази даних MySQL. 2.3 Проєктування підсистеми рекомендацій. 2.4 Проєктування модулів системи. 2.5 UML-моделювання. 2.6 Висновки до другого розділу. 3. Реалізація та експериментальне дослідження. 3.1 Реалізація бази даних MySQL. 3.2 Реалізація вебзастосунку та інтерфейсу користувача на базі фреймворку Laravel. 3.3 Реалізація колаборативної фільтрації. 3.4 Реалізація контентної фільтрації. 3.5 Реалізація гібридної моделі. 3.6 Експериментальне дослідження. 3.7 Аналіз результатів. 3.8 Висновки до розділу. 4. Безпека життєдіяльності, основи охорони праці. 4.1 Питання щодо безпеки життєдіяльності 4.2 Питання з основ охорони праці 4.3 Висновок до четвертого розділу. Висновки. Список використаних джерел. Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., к.т.н, доц. каф. МТ	22.05.2026	29.05.2026

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	27.01.2026-16.02.2026	*Виконано*
3.	Виконання дослідження щодо розробки інтернет-магазину	17.02.2026-24.04.2026	*Виконано*
	Розроблення інтернет-магазину комп'ютерної техніки		
	з підсистемою рекомендації товарів		
4.	Оформлення розділу «Аналіз предметної області	27.04.2026-04.05.2026	*Виконано*
	та методів реалізації рекомендаційних систем»		
5.	Оформлення розділу «Проєктування програмної	05.05.2026-11.05.2026	*Виконано*
	системи»		
6.	Оформлення розділу «Реалізація та експериментальне	12.05.2026-21.05.2026	*Виконано*
	дослідження»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	22.05.2026-25.05.2026	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	26.05.2026-29.05.2026	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	01.06.2026-08.06.2026	*Виконано*
10.	Нормоконтроль	09.06.2026-12.06.2026	*Виконано*
11.	Перевірка на плагіат	15.06.2026	*Виконано*
12.	Попередній захист кваліфікаційної роботи	21.06.2026	*Виконано*
13.	Захист кваліфікаційної роботи	28.06.2026	

Студент _____ Пелішко О.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Палка О.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інтернет-магазину комп’ютерної техніки з підсистемою рекомендації товарів // Кваліфікаційна робота освітнього ступеня «Бакалавр» // Пелішко Олег Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СНс-41 // Тернопіль, 2026 // С. 75, рис. – 16, табл. – 1, додат. – 1, бібліогр. – 30.

Ключові слова: інтернет-магазин, підсистема рекомендацій, колаборативна фільтрація, контентна фільтрація, гібридна модель, фреймворк laravel, база даних mysql, алгоритм k-найближчих сусідів.

Кваліфікаційна робота присвячена дослідженню автоматизованого аналізу споживчого попиту та розробці інтернет-магазину комп’ютерної техніки з підсистемою рекомендацій.

В першому розділі кваліфікаційної роботи описано стан та тенденції розвитку ринку електронної комерції. Висвітлено теоретичні основи побудови інтелектуальних рекомендаційних систем. Розглянуто математичні алгоритми колаборативної та контентної фільтрації.

В другому розділі кваліфікаційної роботи здійснено проєктування архітектури та інформаційного забезпечення вебзастосунку. Досліджено ключові бізнес-процеси та взаємодію компонентів за шаблоном MVC. Подано логічну схему бази даних та прототипи користувацького інтерфейсу.

В третьому розділі кваліфікаційної роботи описано практичну розробку сайту на базі фреймворку Laravel та СУБД MySQL. Проаналізовано програмну реалізацію гібридної моделі на основі алгоритму k-NN та JSON-специфікацій. Проведено експериментальне оцінювання точності рекомендацій.

Об’єкт дослідження: процес функціонування систем електронної комерції та онлайн-продажів комп’ютерної техніки.

Предмет дослідження: моделі, алгоритми та програмні засоби автоматизованого формування гібридних рекомендацій товарів.

ANNOTATION

Development of a Computer Hardware Online Store with a Product Recommendation Subsystem // Qualification work of the educational level «Bachelor» // Pelishko Oleh // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-41 // Ternopil, 2026 // P. 75, fig. – 16, tabl. – 1, annexes. – 1, references – 30.

Keywords: online store, recommendation subsystem, collaborative filtering, content-based filtering, hybrid model, laravel framework, mysql database, k-nearest neighbors algorithm.

The qualification work is dedicated to the research of automated consumer demand analysis and the development of an online computer equipment store with a recommendation subsystem.

The goal of the work is to develop an online computer equipment store with a personalized product recommendation subsystem.

The first section of the qualification paper considered the state of the e-commerce market, the theoretical foundations of intelligent recommendation systems.

In the second section of the qualification work, it is considered the design of the web application architecture, key business processes, the logical database schema, and interface prototypes.

In the third section of the qualification work, the practical development of the website using the Laravel framework and MySQL is described, and the experimental evaluation of recommendation accuracy is conducted.

Object of research: the process of functioning of e-commerce systems and online sales of computer equipment.

Subject of research: models, algorithms, and software tools for automated generation of hybrid product recommendations.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – інтерфейс прикладного програмування для взаємодії між програмними компонентами.

CB (англ. Content-Based) – контентна фільтрація.

CF (англ. Collaborative Filtering) – колаборативна фільтрація.

JSON (англ. JavaScript Object Notation) – текстовий формат обміну даними.

k-NN (англ. k-Nearest Neighbors) – алгоритм k-найближчих сусідів.

MAE (англ. Mean Absolute Error) – середня абсолютна помилка.

MVC (англ. Model-View-Controller) – архітектурний шаблон проєктування програмного забезпечення «Модель-Представлення-Контролер».

MySQL – реляційна система керування базами даних.

PHP (англ. Hypertext Preprocessor) – скриптова мова програмування загального призначення.

SQL (англ. Structured Query Language) – мова структурованих запитів.

UML (англ. Unified Modeling Language) – уніфікована мова моделювання.

URL (англ. Uniform Resource Locator) – єдиний вказівник розміщення веб-ресурсу.

БД – база даних.

ВДТ – відеодисплейний термінал.

ПК – персональний комп’ютер.

СУБД – система управління базами даних.

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок глобалізації, стрімкого розвитку цифрової економіки та масового переходу роздрібної торгівлі в онлайн-простір, ринок електронної комерції зазнає кардинальних трансформацій. Кількість пропозицій на спеціалізованих майданчиках комп'ютерної техніки щодня зростає, що створює для покупців проблему інформаційного перевантаження під час вибору складних технічних комплектуючих. У таких умовах класичні методи пошуку та стандартної навігації каталогом стають недостатньо ефективними. Впровадження інтелектуальних підсистем персоналізованих рекомендацій дозволяє автоматизувати аналіз споживчого попиту, спростити для клієнта процес підбору сумісного заліза та суттєво підвищити конверсію торговельного майданчика, мінімізуючи проблему «холодного старту» для нових товарів. Тому розробка інтернет-магазинів із гібридними рекомендаційними підсистемами є актуальним напрямком сучасних досліджень в галузі комп'ютерних наук та інженерії програмного забезпечення.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення якості послуг електронної комерції та ефективності управління продажами шляхом проектування, розробки та впровадження інтернет-магазину комп'ютерної техніки з інтелектуальною підсистемою генерації персоналізованих рекомендацій товарів.

Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– проаналізувати стан досліджень у сфері електронної комерції та існуючі математичні підходи до побудови рекомендаційних сервісів;

– спроєктувати модульну архітектуру інформаційної системи та реляційну структуру бази даних з оптимізованими JSON-полями для гнучкого збереження специфікацій заліза;

– програмно реалізувати бізнес-логіку вебресурсу на базі сучасного фреймворку Laravel та створити динамічний, ергономічний інтерфейс користувача;

– розробити та інтегрувати підсистему рекомендацій, що базується на гібридній моделі зважування оцінок алгоритму k-найближчих сусідів для колаборативної фільтрації та предикатного контентного аналізу характеристик;

– провести експериментальне дослідження розробленого програмного забезпечення, здійснивши аналіз показників точності та загальної обчислювальної продуктивності системи.

Практичне значення одержаних результатів.

Практичне значення одержаних результатів полягає у створенні повністю працездатного, оптимізованого інтернет-магазину комп’ютерної техніки, який готовий до реального розгортання та комерційної експлуатації. Розроблене програмне забезпечення та інтегрований гібридний рекомендаційний модуль дозволяють в автоматичному режимі аналізувати поведінкові чинники користувачів і специфікації товарів, забезпечуючи високу точність персоналізації при мінімальному часі відгуку сервера. Спроектовані архітектурні рішення, алгоритмічні підходи та розроблені програмні модулі на Laravel мають універсальний характер і можуть бути використані як технологічний базис для масштабування та впровадження аналогічних інтелектуальних сервісів у сучасних системах електронної торгівлі.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА МЕТОДІВ РЕАЛІЗАЦІЇ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1 Аналіз електронної комерції у сфері продажу комп'ютерної техніки

Сучасний стан глобальної економіки характеризується стрімким переходом традиційних форм торгівлі у площину електронної комерції. Сфера продажу комп'ютерної техніки та комплектуючих є одним із найбільш динамічних сегментів цього ринку. Це зумовлено як високим попитом на цифрові рішення для бізнесу та навчання, так і специфікою самих товарів, які легко піддаються стандартизації та каталогізації [15].

Продаж комп'ютерної техніки через інтернет має ряд унікальних характеристик, що суттєво відрізняють цей сегмент від інших категорій товарів, як-от одяг чи продукти харчування:

- технічна складність продукції, оскільки кожна одиниця товару описується десятками критично важливих параметрів, що змушує споживача детально вивчати специфікації перед покупкою;
- проблема взаємної сумісності комплектуючих, яка вимагає від користувача або системи специфічних знань, наприклад, щодо відповідності сокета процесора та роз'єму на материнській платі для забезпечення працездатності майбутньої збірки;
- висока вартість товарів та супутні ризики, що характерні для категорії товарів із високим середнім чеком, де будь-яка помилка у виборі призводить до фінансових втрат та складних логістичних процедур повернення;
- висока динамічність асортименту, зумовлена регулярним оновленням модельних рядів техніки, що вимагає постійної актуалізації бази даних та швидкої адаптації алгоритмів до нових категорій товарів.

Зі зростанням обсягів даних у мережі сучасні інтернет-магазини пропонують тисячі найменувань товарів. Це породжує явище, відоме у маркетингу та когнітивній психології як інформаційне перевантаження [4].

Ілюстрація впливу рекомендаційної системи на воронку продажів зображена на рисунку 1.1.

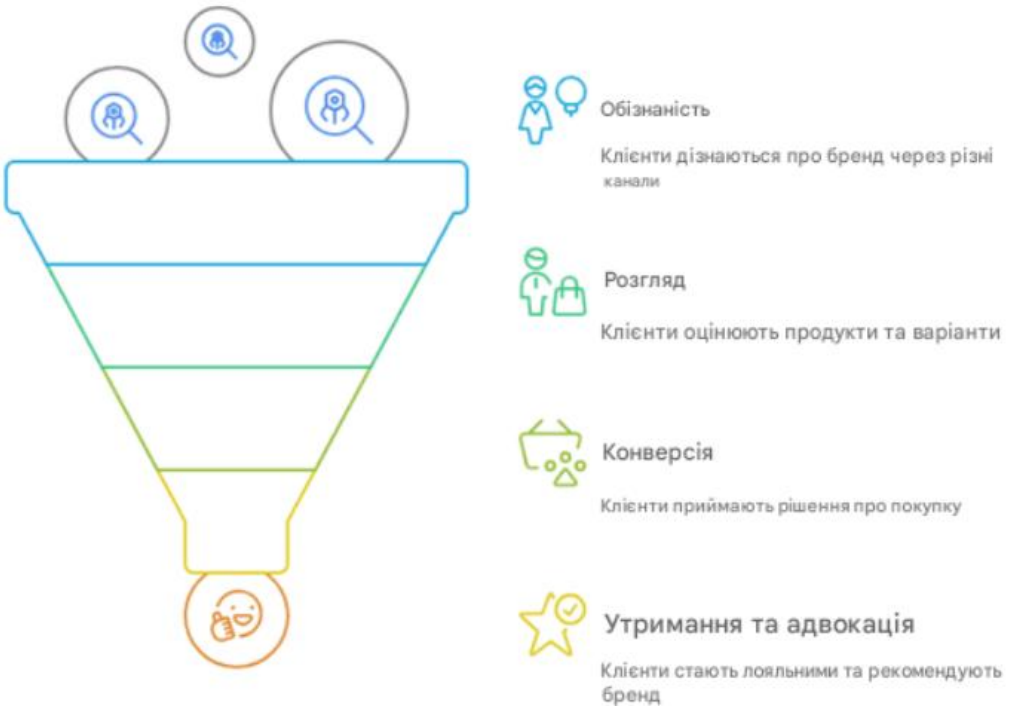

Рисунок 1.1 – Вплив рекомендаційної системи на воронку продажів

Коли користувач стикається з надмірною кількістю варіантів, процес прийняття рішення сповільнюється або стає неможливим. Замість здійснення покупки відвідувач відчуває втому від аналізу даних, що часто призводить до відмови від використання ресурсу. Традиційні інструменти пошуку та фільтрації вже не здатні повноцінно розв’язати цю проблему, оскільки вони потребують від клієнта чіткого розуміння всіх технічних нюансів.

У відповідь на виклики інформаційного перевантаження виникає необхідність впровадження інструментів персоналізації – адаптації контенту сайту під потреби конкретного користувача на основі його поведінки та уподобань.

У сучасних умовах цифрова трансформація торговельних майданчиків та автоматизація аналізу споживчого попиту нерозривно пов’язані із впровадженням аналітики великих даних, яка виступає основою для побудови інтелектуальних систем персоналізації [1].

Впровадження інтелектуальної підсистеми рекомендацій у структуру інтернет-магазину дозволяє ефективно вирішувати наступні завдання:

– максимальне скорочення шляху користувача від входу на сайт до здійснення покупки завдяки пропонуванню найбільш релевантних товарів;

– реалізація механізмів інтелектуального крос-продажу через автоматичне підбирання сумісних аксесуарів та периферійних пристроїв;

– підвищення рівня лояльності споживачів за допомогою створення ефекту «цифрового консультанта», який допомагає орієнтуватися у складному технічному асортименті.

Таким чином, розробка не просто веб-каталогу, а системи з вбудованими механізмами рекомендацій є критично важливою для сучасного конкурентного ринку комп’ютерної техніки.

1.2 Класифікація рекомендаційних систем

Рекомендаційні системи являють собою спеціалізовані алгоритми, призначені для прогнозування ступеня зацікавленості конкретного користувача у певних об’єктах, таких як товари, послуги або інформаційний контент. У сучасній електронній комерції такі системи класифікуються залежно від методів обробки вхідних даних та фундаментальних принципів формування персоналізованих пропозицій [22].

Початковим рівнем надання пропозицій у вебсистемах є нерекомендаційні або статичні підходи, які базуються на узагальненій статистиці магазину та не враховують індивідуальний профіль клієнта:

– формування списків популярних товарів здійснюється на основі аналізу загальної кількості продажів за певний часовий інтервал, що дозволяє виокремити найбільш затребувані позиції каталогу;

– відображення новинок асортименту фокусується на останніх доданих до бази даних товарах, забезпечуючи інформування користувачів про актуальні оновлення модельного ряду;

– виділення акційних пропозицій базується на маркетингових показниках, таких як розмір знижки або необхідність стимулювання попиту на конкретні категорії техніки.

Основним обмеженням таких підходів є відсутність персоналізації, оскільки всі відвідувачі ресурсу отримують ідентичний контент, що суттєво знижує ефективність взаємодії з клієнтами, які мають вузькоспеціалізовані запити.

Даний метод ґрунтується на детальному аналізі характеристик самих об’єктів дослідження. Система формує цифровий профіль кожного товару на основі його технічних атрибутів, що у випадку комп’ютерної техніки включає тип процесора, обсяг оперативної пам’яті, виробника та інші специфікації. Процес рекомендації базується на принципі когнітивної схожості: якщо користувач виявляв зацікавленість у товарі з певними параметрами, алгоритм пропонує йому об’єкти з максимально наближеним вектором характеристик.

Метод колаборативної фільтрації вважається одним із найбільш ефективних підходів, оскільки він спирається не на опис товарів, а на історію поведінки великих груп користувачів. Основна концепція полягає у виявленні патернів схожості між клієнтами: якщо користувачі мають ідентичну історію попередніх взаємодій з асортиментом, то об’єкти, придбані одним із них, з високою ймовірністю зацікавлять інших учасників групи. У межах цього підходу виділяють два ключові вектори аналізу:

– пошук схожих користувачів, де система орієнтується на близькість профілів споживачів;

– пошук схожих товарів, де схожість об’єктів визначається на основі того, наскільки часто вони отримували високі оцінки від одних і тих самих покупців.

Гібридні рекомендаційні системи поєднують у собі переваги кількох методів для досягнення максимальної точності передбачень та мінімізації похибок. Такий підхід дозволяє ефективно нівелювати специфічні недоліки окремих алгоритмів. Зокрема, використання механізмів контентної фільтрації дозволяє рекомендувати нові товари, про які ще немає статистичних даних, у той час як колаборативна складова додає системі здатності знаходити приховані зв’язки між уподобаннями користувачів, які неможливо виявити лише за технічними характеристиками [21].

1.3 Колаборативна фільтрація

Метод колаборативної фільтрації базується на припущенні, що користувачі, які однаково оцінювали певні об’єкти в минулому, схильні демонструвати схожі вподобання і в майбутньому. Основною перевагою цього підходу є його незалежність від характеристик самих товарів, оскільки система аналізує лише патерни взаємодії між суб’єктами та об’єктами.

Фундаментом для роботи алгоритмів колаборативної фільтрації є матриця взаємодій, де рядки відповідають користувачам, а стовпці – товарам. Кожен елемент матриці відображає ступінь зацікавленості користувача у товарі, що може бути виражено через:

– явний зворотний зв’язок, що фіксується у вигляді числових рейтингів або вподобань;

– неявний зворотний зв’язок, який формується на основі аналізу логів поведінки користувача, таких як перегляди карток товарів, додавання у кошик або факт здійснення покупки.

При цьому ефективність функціонування подібних інтелектуальних модулів суттєво залежить від архітектурних рішень щодо збору, фільтрації та пріоритезації обробки даних про реальну поведінку кінцевих користувачів [2].

У реальних умовах інтернет-магазину така матриця є надзвичайно розрідженою, оскільки кожен окремий клієнт взаємодіє лише з мізерною часткою загального асортименту техніки.

Для формування рекомендацій у межах колаборативної моделі використовують два основні підходи, кожен з яких має свою специфіку застосування:

– підхід, орієнтований на користувачів, передбачає пошук «сусідів» для цільового користувача – людей зі схожою історією взаємодій, на основі вподобань яких формується список пропозицій;

– підхід, орієнтований на товари, фокусується на обчисленні подібності між самими об’єктами на основі того, наскільки часто їх купували або оцінювали одні й ті самі люди. Для магазинів комп’ютерної техніки з великою кількістю користувачів та відносно стабільним каталогом товарів підхід Item-Based часто є більш стабільним та обчислювально ефективним [27].

Для визначення ступеня близькості між двома векторами (користувачів або товарів) найчастіше використовується косинусна міра схожості. Вона дозволяє оцінити кут між векторами у багатовимірному просторі, що є ефективнішим за звичайну евклідову відстань у задачах рекомендацій, оскільки орієнтується на напрямок вподобань, а не на їх абсолютну величину.

Формула (1.1) косинусної схожості між векторами A та B.

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\| A \| \cdot \| B \|} = \frac{\sum_{i=1}^{n} a_i \, b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}} \qquad (1.1)$$

Де значення результату лежить у діапазоні від -1 до 1, де 1 означає повну ідентичність векторів, а 0 – відсутність будь-якої кореляції між ними.

Незважаючи на високу точність у виявленні неочевидних зв’язків, метод має певні обмеження, які необхідно враховувати при проектуванні системи:

– до переваг належить здатність системи адаптуватися до складних уподобань користувачів без необхідності детального опису кожного товару, що дозволяє знаходити цікаві пропозиції у суміжних категоріях техніки;

– основним недоліком є обчислювальна складність при масштабуванні системи, оскільки розрахунок схожості для мільйонів користувачів вимагає значних серверних ресурсів;

– проблема «холодного старту» виникає у ситуаціях, коли до системи додається новий користувач або новий товар, про які ще немає даних у матриці взаємодій, що робить неможливим формування коректних рекомендацій колаборативним методом.

Таким чином, колаборативна фільтрація виступає потужним інструментом для виявлення прихованих закономірностей у вподобаннях користувачів, забезпечуючи високу релевантність пропозицій для клієнтів із насиченою історією взаємодій. Проте, враховуючи специфіку магазину комп’ютерної техніки, де асортимент оновлюється надзвичайно швидко, критично важливо знайти рішення для проблеми «холодного старту» нових комплектуючих. Це зумовлює необхідність залучення методів, які базуються не на поведінці користувачів, а на аналізі безпосередніх технічних характеристик самих товарів, що і буде детально розглянуто у наступному підрозділі.

1.4 Контентна фільтрація

Контентна фільтрація базується на аналізі внутрішніх властивостей об’єктів та формуванні рекомендацій на основі порівняння характеристик товарів із профілем уподобань користувача. На відміну від колаборативного підходу, цей метод дозволяє ефективно працювати з новими товарами, для яких ще не накопичено статистику продажів, що є критично важливим для ринку комп’ютерної техніки, який постійно оновлюється.

Для того, щоб алгоритм міг порівнювати товари між собою, кожен об’єкт у базі даних необхідно представити у вигляді числового вектора. У випадку інтернет-магазину техніки цей процес передбачає перетворення текстових та категоріальних описів (наприклад, тип сокета, модель чипсета або бренд) у набір числових значень. Кожна характеристика стає окремим виміром у багатовимірному просторі ознак, що дозволяє системі математично обчислювати відстань або близькість між різними комплектуючими.

Оскільки більшість технічних характеристик є категоріальними (наприклад, назва виробника: ASUS, MSI, Gigabyte), для їх обробки застосовується метод One-hot encoding. Суть методу полягає у створенні бінарного вектора для кожної категорії, де кожному унікальному значенню відповідає окремий стовпець. Значення «1» ставиться у позиції, що відповідає поточній характеристиці, а всі інші позиції заповнюються нулями. Це дозволяє уникнути помилкового ранжування категорій, коли система могла б сприйняти бренд №2 як «кращий» за бренд №1 лише через його порядковий номер і забезпечує коректну роботу математичних моделей.

Окрім категоріальних даних, технічні специфікації містять велику кількість числових параметрів, таких як тактова частота процесора, обсяг накопичувача або ціна. Оскільки ці значення мають різні діапазони (наприклад, ціна може вимірюватися десятками тисяч, а кількість ядер – одиницями), виникає потреба у нормалізації. Найчастіше використовується метод Min-Max нормалізації, який приводить усі числові значення до єдиного діапазону [0, 1]. Це гарантує, що параметри з великими абсолютними значеннями (ціна) не будуть домінувати над іншими важливими характеристиками при обчисленні загальної схожості товарів.

Після векторизації та нормалізації всіх ознак система обчислює ступінь схожості між поточним товаром, який переглядає користувач, та іншими позиціями в каталозі. Для цього використовується косинусна міра схожості (аналогічна описаній у підрозділі 1.3), проте в даному контексті векторами виступають самі характеристики товарів. Чим ближчим є значення косинуса до

одиниці, тим більше схожих параметрів мають комп’ютерні компоненти, що дозволяє системі пропонувати користувачеві максимально релевантні альтернативи або доповнення до вибраної позиції.

Застосування контентної фільтрації дозволяє побудувати надійну систему рекомендацій, яка забезпечує високу точність на основі об’єктивних даних про товар. Такий підхід робить систему стійкою до дефіциту даних про нових клієнтів, проте для досягнення максимальної ефективності та врахування складних поведінкових чинників отримані результати доцільно об’єднувати з методами інтелектуального пошуку сусідів, що будуть розглянуті далі.

1.5 Алгоритм k-найближчих сусідів

Алгоритм k-найближчих сусідів належить до методів навчання на основі прикладів і є одним із найбільш розповсюджених підходів у побудові рекомендаційних систем. Основна ідея алгоритму в контексті інтернет-магазину полягає у виявленні певної кількості k найбільш схожих об’єктів або користувачів до цільового елемента на основі обраної метрики відстані [20].

Робота алгоритму kNN базується на геометричному представленні даних у багатовимірному просторі ознак. Після того, як товари були перетворені у вектори, система розраховує відстань між вектором цільового товару та всіма іншими векторами у базі даних. Найчастіше для цього використовується косинусна схожість або евклідова відстань. На основі отриманих результатів система ранжує всі об’єкти та відбирає k елементів із найменшою відстанню, які і стають основою для формування блоку рекомендацій «Схожі товари» або «З цим також купують».

Принцип пошуку найближчих сусідів у багатовимірному просторі ознак продемонстровано на рисунку 1.2.

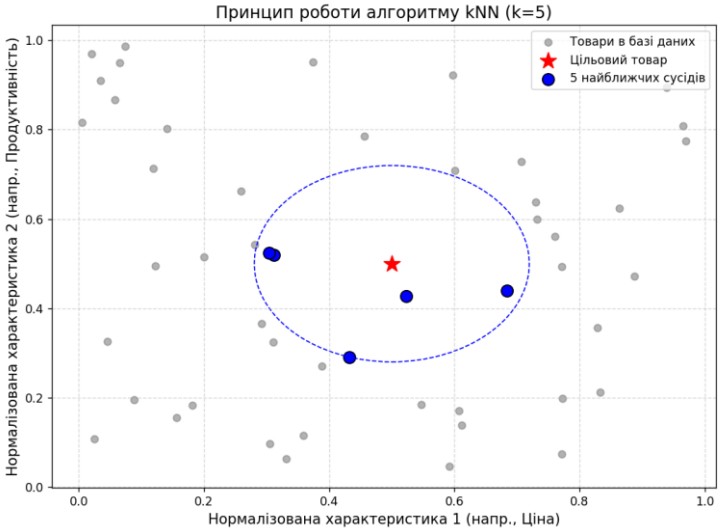

Рисунок 1.2 – Візуалізація пошуку k найближчих сусідів для цільового товару

Ключовим фактором, що впливає на якість роботи алгоритму, є вибір значення гіперпараметра k, який визначає кількість сусідів для аналізу. Вибір занадто малого значення може призвести до надмірної чутливості системи до шумів у даних та аномальних вподобань окремих користувачів. З іншого боку, занадто велике значення k призводить до «розмивання» рекомендацій, оскільки система починає враховувати об’єкти, які мають лише посередню схожість із цільовим товаром. У практичній реалізації систем електронної комерції оптимальне значення k зазвичай підбирається експериментально шляхом аналізу метрик точності на тестових даних.

Однією з головних особливостей алгоритму kNN є те, що він не потребує етапу попереднього навчання моделі, проте всі обчислення відбуваються безпосередньо під час запиту. Це зумовлює певну обчислювальну складність, оскільки для пошуку сусідів необхідно порівняти цільовий вектор із кожним об’єктом у базі даних, що має складність порядку O (n * d), де n – кількість товарів, а d – кількість характеристик [9]. Для великих інтернет-магазинів із десятками тисяч комплектуючих це може створювати затримки у відгуку інтерфейсу, що вимагає впровадження методів оптимізації, таких як індексування простору або використання алгоритмів наближеного пошуку найближчих сусідів.

1.6 Гібридна модель рекомендацій

На основі аналізу методів колаборативної та контентної фільтрації можна зробити висновок, що жоден із них не є універсальним рішенням для сучасного магазину комп'ютерної техніки. Колаборативна фільтрація демонструє високу точність на великих обсягах даних про поведінку, але потерпає від проблеми «холодного старту», тоді як контентна фільтрація ефективно працює з новинками, але часто обмежує вибір лише дуже схожими об'єктами. Для подолання цих недоліків у даній роботі пропонується використання гібридної моделі рекомендацій.

Гібридна модель у межах розроблюваної системи базується на зваженому поєднанні оцінок, отриманих від двох різних підсистем. Математично таку модель можна представити у вигляді лінійної комбінації (1.2).

$$Score = \alpha \cdot CF + \beta \cdot CB, \tag{1.2}$$

де компоненти формули мають наступне значення:

- CF – нормалізована оцінка релевантності товару, розрахована на основі подібності вподобань групи користувачів;
- CB – нормалізована оцінка схожості товару, розрахована на основі аналізу технічних характеристик комплектуючих;
- α та β – вагові коефіцієнти, що визначають ступінь впливу кожної моделі на підсумковий результат.

Обов'язковою умовою для коректної роботи системи є дотримання рівності α + β = 1. Це дозволяє гнучко налаштовувати систему залежно від стадії життєвого циклу магазину або конкретних бізнес-завдань.

Перевага гібридної моделі полягає у її здатності забезпечувати релевантні рекомендації у будь-яких умовах. Якщо у системі з'являється новий товар, про який ще немає даних у матриці взаємодій, коефіцієнт β дозволяє системі

рекомендувати його користувачам на основі технічних специфікацій. У міру накопичення статистики покупок та оцінок, вага колаборативної складової α може бути збільшена, що дозволить системі виявляти глибші поведінкові зв’язки та пропонувати несподівані, але доречні товари.

Таким чином, використання гібридної моделі дозволяє максимізувати точність системи персоналізації, забезпечуючи при цьому стійкість до проблеми дефіциту даних. Поєднання математичної точності контентного аналізу з гнучкістю колаборативної фільтрації створює надійну базу для розробки інтелектуального модуля рекомендацій. Для підтвердження ефективності такого підходу на практиці необхідно визначити набір метрик, за якими буде оцінюватися якість сформованих пропозицій.

1.7 Метрики оцінювання рекомендацій

Для визначення ефективності розроблених алгоритмів та вибору оптимальних параметрів гібридної моделі необхідно використовувати кількісні показники якості. Оскільки рекомендаційні системи за своєю природою є імовірнісними, їх оцінювання базується на порівнянні сформованих пропозицій із реальними вподобаннями користувачів, зафіксованими у тестовій вибірці даних.

У системах електронної комерції найчастіше оцінюється не вся видача, а лише перші K об’єктів, які бачить користувач. Для цього застосовуються наступні метрики:

- точність на рівні K визначає частку релевантних товарів серед перших K запропонованих системою. Вона розраховується як відношення кількості рекомендованих товарів, що зацікавили користувача, до загальної кількості наданих рекомендацій K;
- повнота на рівні K відображає здатність системи знайти всі релевантні товари з тестового набору даних. Метрика розраховується як відношення

кількості знайдених релевантних товарів у списку Top-K до загальної кількості товарів, які користувач реально придбав або оцінив позитивно.

Для Precision@K формулу (1.3) можна представити наступним чином [17].

$$Precision@K = \frac{Relevant\_Items \cap Recommended\_Items@K}{K} \quad (1.3)$$

Візуальне представлення базових показників, що використовуються для розрахунку метрик точності, наведено у вигляді матриці помилок на рис. 1.3.

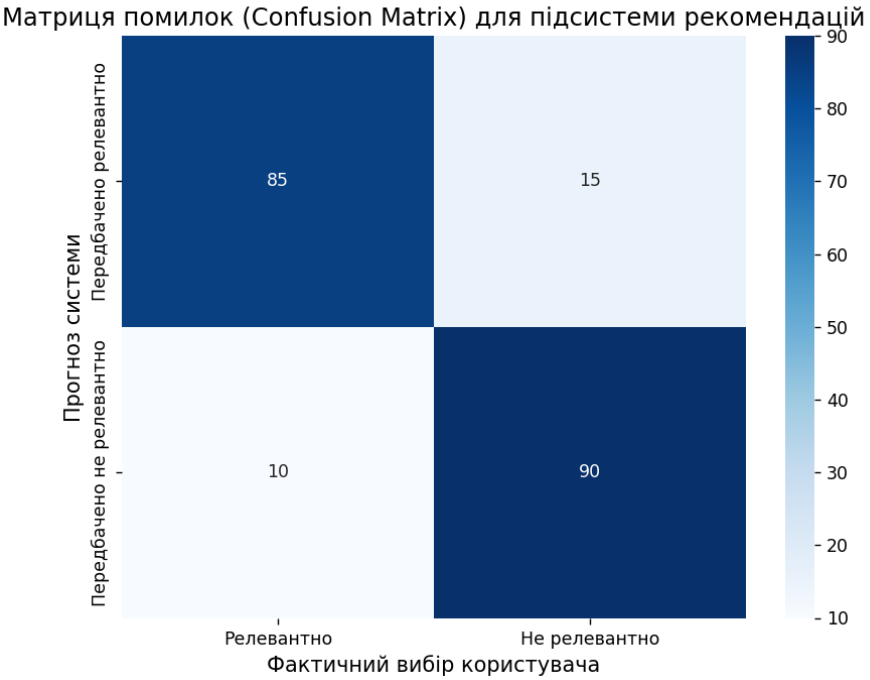

Рисунок 1.3 – Матриця помилок для оцінювання релевантності рекомендацій

Оскільки Precision та Recall часто знаходяться у зворотній залежності, для комплексної оцінки використовується F1-score. Це середнє гармонійне між точністю та повнотою, яке дозволяє отримати єдине числове значення ефективності алгоритму. Високе значення F1-score свідчить про те, що система знаходить значну частину релевантних товарів, зберігаючи при цьому високу якість кожної окремої рекомендації (1.4).

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (1.4)$$

У випадках, коли система працює з явними рейтингами, використовується метрика MAE. Вона вимірює середнє відхилення передбаченої системою оцінки від фактичної оцінки, яку поставив користувач. Чим нижчим є значення MAE, тим точніше модель прогнозує вподобання клієнта.

Використання зазначеного набору метрик дозволяє об’єктивно порівняти різні конфігурації гібридної системи та обрати вагові коефіцієнти, що забезпечать найкращий досвід користувача. Обґрунтування обраних методів та визначення цільових показників точності дозволяє перейти до безпосереднього формулювання вимог до розробки програмного продукту.

1.8 Постановка задачі

На основі проведеного аналізу предметної області, вивчення існуючих методів фільтрації та принципів побудови гібридних рекомендаційних систем, можна сформулювати основну мету та перелік завдань кваліфікаційної роботи.

Метою роботи є розробка веб-орієнтованої інформаційної системи інтернет-магазину комп’ютерної техніки, оснащеної інтелектуальною підсистемою рекомендацій, що дозволяє вирішити проблему інформаційного перевантаження та підвищити релевантність пропозицій для користувачів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Спроектувати архітектуру бази даних у середовищі MySQL, яка б забезпечувала надійне зберігання не лише інформації про товари та замовлення, а й логів взаємодії користувачів для роботи алгоритмів рекомендацій [7].
2. Розробити веб-застосунок на базі сучасного фреймворку Laravel, реалізувавши основні функції електронної комерції: каталог товарів, систему фільтрації, кошик та кабінет користувача.

3. Реалізувати модуль контентної фільтрації, що базується на векторизації технічних характеристик комп’ютерних комплектуючих та обчисленні косинусної схожості між ними.

4. Розробити механізм колаборативної фільтрації, використовуючи алгоритм k-найближчих сусідів для виявлення патернів схожості між вподобаннями різних клієнтів.

5. Інтегрувати гібридну модель, що поєднує результати контентного та колаборативного аналізу, забезпечуючи стійкість системи до проблеми холодного старту.

6. Провести експериментальну перевірку розробленої системи на тестовому наборі даних, оцінивши її ефективність за допомогою метрик Precision@K, Recall@K та F1-score.

1.9 Висновки до першого розділу

У першому розділі кваліфікаційної роботи було здійснено комплексний теоретико-методологічний та аналітичний огляд сучасного стану електронної комерції у сегменті роздрібного продажу комп’ютерної техніки та комплектуючих. Досліджено фундаментальні принципи функціонування сучасних рекомендаційних систем, проаналізовано їхні математичні моделі та архітектурні особливості, а також науково обґрунтовано доцільність впровадження інтелектуальних методів персоналізації. На основі виявлених переваг та обмежень існуючих підходів сформовано базис для подальшого проектування та практичної реалізації програмного продукту.

За результатами проведеного аналізу предметної області та дослідження математичного апарату було отримано такі науково-практичні висновки:

– вивчення специфіки ринку електронної комерції у ніші комп’ютерної техніки дозволило виявити критичну проблему інформаційного перевантаження користувачів. Через високу технічну складність продукції, значну кількість специфічних характеристик комплектуючих та проблему

їхньої взаємної сумісності, традиційні статичні інструменти пошуку та фільтрації виявляються неефективними, що зумовлює гостру необхідність інтеграції інтелектуальних підсистем персоналізації як цифрових консультантів;

– критичний аналіз класичних підходів до побудови рекомендаційних систем продемонстрував, що ізольоване використання контентної або колаборативної фільтрації не здатне забезпечити стабільну якість рекомендацій. Колаборативний підхід має високу точність на етапі зрілості системи, але критично залежить від щільності матриці взаємодій, тоді як контентна фільтрація ефективно працює за умов відсутності поведінкової статистики, проте обмежена виключно подібними за характеристиками товарами, що звужує різноманітність видачі;

– обґрунтовано вибір алгоритму k-найближчих сусідів як математичного ядра для пошуку релевантних пропозицій у метричному просторі ознак. Визначено, що для коректної роботи алгоритму категоріальні технічні специфікації комплектуючих мають піддаватися попередній векторизації за технологією One-hot encoding, а числові параметри – Min-Max нормалізації, що запобігає домінуванню ознак із великими абсолютними значеннями;

– спроектовано структуру гібридної моделі рекомендацій, яка базується на зваженій лінійній комбінації оцінок контентного та колаборативного модулів із використанням гнучких вагових коефіцієнтів α та β за умови дотримання рівності їхньої суми одиниці. Такий підхід забезпечує високу адаптивність інформаційної системи, дозволяючи динамічно нівелювати проблему холодного старту для нових товарів та користувачів шляхом автоматичного перерозподілу пріоритету між алгоритмами;

– сформовано комплексну методологію оцінювання ефективності розроблюваних алгоритмів на основі імовірнісних метрик якості. Для аналізу точності списку пропозицій на рівні Top-K обрано метрики Precision@K та Recall@K, а для отримання інтегральної оцінки балансу між ними – показник F1-score;

– формалізовано чітку постановку задачі на наступні етапи дослідження, яка передбачає проектування реляційної бази даних у СКБД MySQL та розробку веб-орієнтованого застосунку на базі сучасного фреймворку Laravel з використанням архітектурного шаблону MVC.

Узагальнюючи викладене, результати першого розділу сформували цілісну теоретичну та алгоритмічну платформу.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Загальна архітектура системи

Проєктування сучасної інформаційної системи електронної комерції вимагає використання архітектурних підходів, що забезпечують високу швидкість обробки запитів, масштабованість, безпеку даних та незалежність окремих модулів. Для реалізації інтернет-магазину комп'ютерної техніки з інтегрованою підсистемою рекомендацій було обрано класичну трирівневу архітектуру.

Цей підхід передбачає чіткий поділ системи на три взаємопов'язані, але незалежні рівні, що дозволяє відокремити інтерфейс користувача від бізнес-логіки та механізмів збереження даних [8].

Рівень представлення є верхнім шаром архітектури, з яким безпосередньо взаємодіє кінцевий користувач системи. На відміну від застарілих десктопних рішень, сучасна реалізація інтернет-магазину базується на веб-орієнтованому інтерфейсі, доступному через будь-який сучасний браузер. Клієнтська частина виконує такі функції:

– відображення динамічного контенту (каталогу комп'ютерної техніки, карток товарів, персоналізованих рекомендацій);

– збір та відправка даних користувача на сервер (процеси реєстрації, авторизації, оформлення замовлення);

– первинна валідація введених форм на стороні клієнта за допомогою мови програмування JavaScript;

– взаємодія з серверною частиною за допомогою асинхронних запитів для оновлення елементів сторінки (наприклад, додавання товару в кошик) без повного перезавантаження сторінки.

Центральним елементом архітектури є серверний рівень, де зосереджена вся бізнес-логіка інтернет-магазину. Реалізація цього рівня здійснюється на базі

сучасного PHP-фреймворку Laravel, який використовує архітектурний шаблон MVC [5]. Бізнес-рівень відповідає за:

– маршрутизацію запитів користувачів та забезпечення безпеки (захист від CSRF-атак, SQL-ін’єкцій, XSS-вразливостей);

– автентифікацію та авторизацію користувачів із розмежуванням прав доступу (клієнт, менеджер, адміністратор);

– керування кошиком, обробку транзакцій, розрахунок підсумкової вартості замовлень та взаємодію із зовнішніми API;

– підготовку та агрегацію даних, що передаються з рівня бази даних до клієнтського застосунку.

Нижній рівень архітектури забезпечує довготривале та надійне збереження всієї інформації застосунку. Для цього використовується реляційна система керування базами даних MySQL [24]. Рівень даних забезпечує виконання CRUD-операцій над сутностями системи, гарантує цілісність даних за допомогою зовнішніх ключів та транзакцій, а також оптимізує швидкість вибірки великих масивів інформації.

Підсистема рекомендацій інтегрується у бізнес-рівень системи як окремий інтелектуальний модуль. Вона функціонує у тісному зв’язку з рівнем збереження даних та логікою Laravel. Принцип її інтеграції полягає в наступному: модуль рекомендацій перехоплює дії користувача, записує їх у БД, а під час формування веб-сторінки надсилає запит до бази даних для отримання векторів характеристик товарів та матриці взаємодій. Проводячи математичні розрахунки за гібридною моделлю, підсистема миттєво повертає масив ідентифікаторів релевантних товарів у контролер Laravel, який передає їх на клієнтський застосунок для відображення споживачу.

Загальну схему трирівневої архітектури розроблюваної інформаційної системи та взаємодію її компонентів наведено на рис. 2.1.

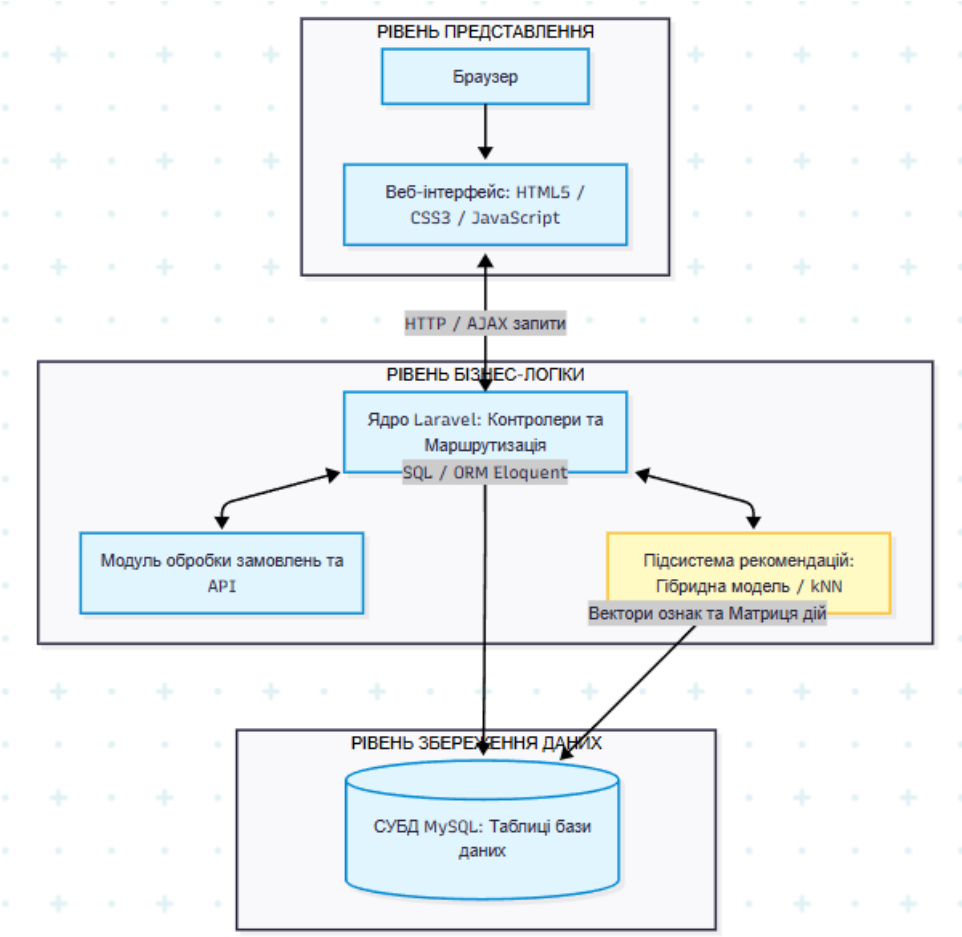

Рисунок 2.1 – Схема трирівневої архітектури інформаційної системи інтернет-магазину

Така організація архітектури дозволяє ізолювати обчислювально складні процеси підсистеми рекомендацій та аналізу споживчого попиту від інтерфейсу користувача. Це гарантує, що проведення математичних розрахунків гібридної моделі не впливатиме на швидкість завантаження сторінок інтернет-магазину та загальний рівень чуйності інтерфейсу. Таким чином, обрана трирівнева структура забезпечує високу відмовостійкість, гнучкість у масштабуванні окремих модулів системи та створює надійну основу для подальшої програмної реалізації бази даних і серверних контролерів.

2.2 Проєктування бази даних MySQL

Проєктування реляційної бази даних є одним із найважливіших етапів розробки системи, оскільки від структури таблиць та зв'язків між ними залежить швидкість роботи сайту та точність рекомендаційних алгоритмів. Для функціонування інтернет-магазину комп'ютерної техніки було спроєктовано оптимальну логічну структуру бази даних, яка складається з шести основних таблиць [26].

Таблиця Users призначена для збереження облікових даних користувачів системи. Вона містить інформацію, необхідну для авторизації та розрахунку колаборативних рекомендацій:

- id (INT, Primary Key, Auto Increment) – унікальний ідентифікатор користувача;
- name (VARCHAR) – ім'я та прізвище користувача;
- email (VARCHAR, Unique) – електронна адреса (використовується як логін);
- password (VARCHAR) – хешований пароль;
- role (ENUM) – роль у системі (client, admin);
- created_at, updated_at (TIMESTAMP) – системні часові мітки фреймворку Laravel.

Таблиця Categories зберігає ієрархічну структуру категорій комп'ютерної техніки (наприклад: процесори, відеокарти, оперативна пам'ять):

- id (INT, Primary Key, Auto Increment) – унікальний ідентифікатор категорії;
- name (VARCHAR) – назва категорії;
- slug (VARCHAR, Unique) – людино-зрозумілий URL для категорії;
- parent_id (INT, Nullable) – зовнішній ключ на цю ж таблицю для реалізації підкатегорій.

Таблиця Products містить інформацію про комп’ютерну техніку та комплектуючі. Специфікації товарів зберігаються у форматі JSON, що дозволяє гнучко описувати різні за структурою параметри комплектуючих для подальшої векторизації та контентної фільтрації [13]:

- id (INT, Primary Key, Auto Increment) – унікальний ідентифікатор товару;
- category_id (INT, Foreign Key) – зв’язок із таблицею категорій;
- name (VARCHAR) – повна назва товару;
- description (TEXT) – детальний опис комплектуючого;
- price (DECIMAL) – вартість товару;
- stock (INT) – кількість одиниць на складі;
- specifications (JSON) – набір технічних характеристик (сокет, частота, TDP тощо);
- created_at, updated_at (TIMESTAMP) – часові мітки.

Таблиця Orders зберігає загальну інформацію про замовлення, здійснені клієнтами магазину:

- id (INT, Primary Key, Auto Increment) – унікальний ідентифікатор замовлення;
- user_id (INT, Foreign Key) – зв’язок із користувачем, який зробив замовлення;
- total_price (DECIMAL) – загальна сума замовлення;
- status (ENUM) – статус замовлення (new, processing, shipped, completed, canceled);
- delivery_info (TEXT) – дані для доставки;
- created_at, updated_at (TIMESTAMP) – часові мітки.

Таблиця OrderItems є сполучною таблицею між замовленнями та товарами, фіксуючи, які саме товари та в якій кількості увійшли в конкретний чек:

- id (INT, Primary Key, Auto Increment) – унікальний ідентифікатор запису;
- order_id (INT, Foreign Key) – зв’язок із таблицею замовлень;
- product_id (INT, Foreign Key) – зв’язок із таблицею товарів;
- quantity (INT) – кількість замовлених одиниць товару;
- price (DECIMAL) – ціна товару на момент покупки.

Таблиця UserActions критично важлива таблиця для побудови матриці взаємодій підсистеми рекомендацій. Вона фіксує всі неявні сигнали інтересу користувача до товарів.

- id (INT, Primary Key, Auto Increment) – унікальний ідентифікатор дії;
- user_id (INT, Foreign Key) – ідентифікатор користувача;
- product_id (INT, Foreign Key) – ідентифікатор товару;
- action_type (ENUM) – тип дії користувача (view – перегляд, cart – додавання в кошик, purchase – покупка);
- weight (INT) – числове значення ваги дії (наприклад: view = 1, cart = 3, purchase = 5), що використовується для заповнення матриці взаємодій;
- timestamp (TIMESTAMP) – час здійснення дії.

Для відображення логічної структури бази даних, визначення первинних і зовнішніх ключів, а також встановлення типів зв’язків між сутностями розробляється ER-діаграма.

Графічне представлення структури зв’язків спроєктованої бази даних інтернет-магазину наведено на рис. 2.2.

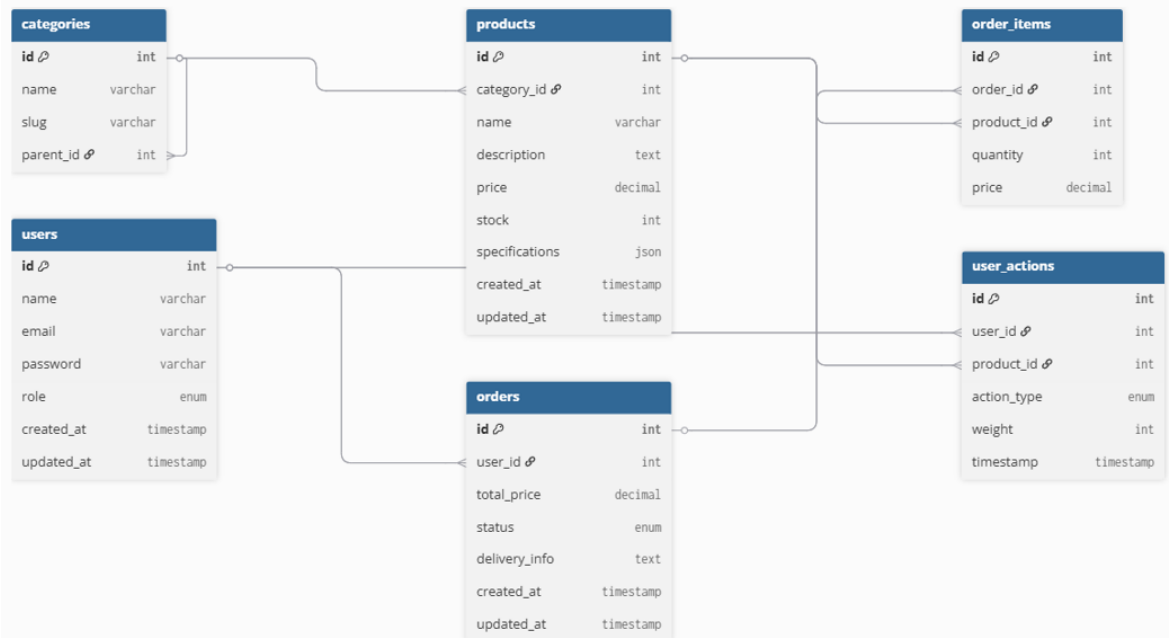

Рисунок 2.2 – ER-діаграма бази даних інформаційної системи

Для мінімізації надлишковості даних, запобігання аномаліям модифікації, видалення та вставки, а також для забезпечення оптимальної продуктивності підсистеми рекомендацій, спроєктована база даних була піддана процедурі нормалізації до третьої нормальної форми [18].

Перша нормальна форма вимагає, щоб усі атрибути таблиць були атомарними, а кожна стрічка була унікальною. Спроєктована БД відповідає 1NF, оскільки всі текстові та числові поля містять лише одне значення, а унікальність забезпечується впровадженням первинних ключів для кожного запису. Поле specifications у форматі JSON концептуально є атомарним для СКБД, оскільки розглядається як єдиний об’єкт властивостей конкретного екземпляра техніки.

Друга нормальна форма потребує виконання умов 1NF та відсутності часткових залежностей неключових атрибутів від складеного первинного ключа. Оскільки в усіх основних таблицях як первинний ключ використано сурогатний простий ключ id, будь-яка часткова залежність виключається автоматично. Проміжна таблиця OrderItems залежить від свого власного id, а зв’язки базуються на повних зовнішніх ключах, що повністю задовольняє вимогам 2NF.

Третя нормальна форма вимагає виконання умов 2NF та відсутності транзитивних залежностей неключових атрибутів від первинного ключа. У спроєктованій структурі бази даних цей принцип чітко витримано. Наприклад, у таблиці Orders зберігається лише ідентифікатор користувача user_id, а його ім’я чи email не дублюються, оскільки вони залежать від первинного ключа таблиці Users. Ціна у таблиці OrderItems відображає фіксований історичний момент транзакції і не є транзитивною залежністю від Products.price.

2.3 Проєктування підсистеми рекомендацій

Проєктування підсистеми рекомендацій вимагає детального алгоритмічного опису процесів збору, обробки та перетворення даних, що надходять із бази даних MySQL, у кінцевий персоналізований список товарів. Оскільки архітектура системи спирається на гібридну модель, алгоритм функціонує як послідовний конвеєр обчислень.

Першим етапом є вибірка логів із таблиці UserActions. Система агрегує всі дії поточного користувача та інших клієнтів і трансформує їх у числовий масив – розріджену матрицю взаємодій. Кожному типу дії присвоюється статична вага, що дозволяє перетворити неявні сигнали поведінки у сурогатні рейтинги.

Паралельно з формуванням матриці дій підсистема звертається до таблиці Products для вилучення технічних специфікацій комплектуючих. Текстові та категоріальні поля проходять процес бінаризації за допомогою One-hot encoding, а числові параметри нормалізуються у діапазоні [0, 1]. Результатом цього етапу є створення щільних числових векторів фіксованої розмірності для кожної одиниці комп’ютерної техніки.

На основі побудованих векторів ознак та векторів оцінок підсистема запускає цикл обчислення косинусної схожості. Алгоритм розраховує скалярний добуток векторів, поділений на добуток їхніх евклідових норм, що

дозволяє отримати числовий коефіцієнт близькості між цільовим товаром та іншими позиціями каталогу.

Отримані коефіцієнти схожості піддаються сортуванню у порядку спадання. Алгоритм k-NN відсікає всі об’єкти, що знаходяться поза межами встановленого гіперпараметра k, формуючи робочу множину найближчих сусідів для подальшого аналізу.

На фінальному етапі відібрані кандидати проходять процедуру зваженого злиття оцінок. Система застосовує лінійну формулу (2.1).

$$Score = \alpha * CF + \beta * CB \quad (2.1)$$

Якщо користувач є новим, вага колаборативної оцінки α прирівнюється до нуля, і рейтинг формується виключно на основі контентної схожості CB. Для активних користувачів система балансує обидва показники, ранжує товари за фінальним балом Score і передає топ-результати у контролер Laravel для виведення на сторінку.

Для візуалізації послідовності виконання операцій та керування потоками даних розроблено блок-схему алгоритму гібридної рекомендаційної системи (див. рисунок 2.3).

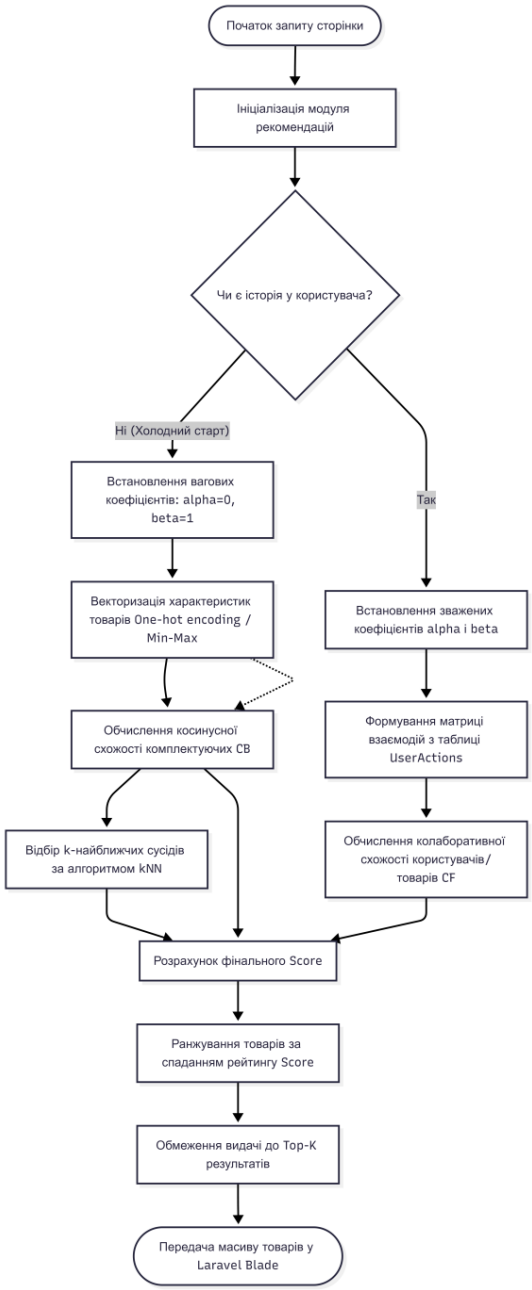

Рисунок 2.3 – Блок-схема алгоритму функціонування гібридної підсистеми рекомендацій

Таким чином, розроблена блок-схема алгоритму чітко регламентує послідовність процесів обробки даних та формалізує логіку ухвалення рішень підсистемою рекомендацій у різних сценаріях взаємодії з клієнтами. Представлена архітектурна модель дозволяє детально візуалізувати потоки інформації між контентним і колаборативним модулями, забезпечуючи наочне рішення для подолання проблеми холодного старту. Описана алгоритмічна

логіка є центральним інтелектуальним ядром застосунку, яке безпосередньо інтегрується в загальну структуру вебресурсу та взаємодіє з іншими функціональними компонентами інтернет-магазину, детальне проєктування яких наведено у наступному підрозділі.

2.4 Проєктування модулів системи

Інформаційна система інтернет-магазину комп’ютерної техніки будується за модульним принципом на основі архітектурного шаблону MVC фреймворку Laravel [3]. Це забезпечує слабку зв’язаність компонентів та дозволяє ізольовано розробляти і тестувати окремі функціональні блоки сайту.

Модуль авторизації та автентифікації забезпечує безпечний доступ користувачів до персоналізованих функцій ресурсу. Він реалізує логіку реєстрації нових клієнтів, перевірки облікових даних при вході, шифрування паролів за допомогою алгоритму bcrypt та керування сесіями. Завдяки інтеграції механізмів Middleware у Laravel, модуль автоматично розмежовує права доступу до адміністративної частини сайту та особистого кабінету покупця.

Модуль каталогу товарів відповідає за структурування, фільтрацію та відображення комп’ютерних комплектуючих. Він обробляє запити до бази даних через ORM Eloquent, реалізує динамічне розбиття на сторінки та підтримує складні багатофакторні фільтри за технічними параметрами, які передаються через URL-параметри.

Модуль кошика та обробки замовлень забезпечує бізнес-логіку купівлі товарів [19]. Він дозволяє користувачеві тимчасово зберігати обрані позиції техніки в сесії або базі даних, динамічно перераховувати загальну вартість при зміні кількості одиниць, перевіряти залишки товарів на складі та трансформувати вміст кошика у замовлення з присвоєнням початкового статусу у таблиці Orders.

Модуль адміністрування призначений для менеджерів та адміністраторів сайту. Він надає графічний інтерфейс для виконання CRUD-операцій над каталогом таких як додавання нових моделей відеокарт, редагування цін, зміна опису, тощо.

Модуль рекомендацій є інтелектуальною надбудовою системи, яка інтегрується у контролери каталогу та картки товару. Він асинхронно фіксує події користувача для оновлення таблиці дій, викликає обчислювальні скрипти гібридної фільтрації та генерує персоналізовані блоки «Рекомендовано для вас» на головній сторінці або «З цим товаром купують сумісні комплектуючі» у картці товару [23].

2.5 UML-моделювання

Для детального проєктування функціональних вимог, внутрішньої структури та динаміки взаємодії компонентів інтернет-магазину комп'ютерної техніки застосовано методологію об'єктно-орієнтованого аналізу із використанням уніфікованої мови моделювання UML. Це дозволяє створити несуперечливу візуальну модель системи, що поєднує поведінковий та структурний аспекти програмного продукту.

Діаграма прецедентів (див. рисунок 2.4) визначає функціональні межі системи та моделює взаємодію між зовнішніми суб'єктами і ключовими сценаріями використання вебресурсу. На відміну від базових рішень, у розроблюваній системі реалізовано трирольову модель взаємодії, яка чітко розмежовує можливості анонімних відвідувачів, постійних клієнтів та адміністратора.

Незареєстрований користувач – базовий актор системи, який взаємодіє з торговельним майданчиком анонімно. Для нього доступні прецеденти, пов'язані з ознайомленням із асортиментом та базовими комерційними діями: «перегляд каталогу товарів», «перегляд технічних специфікацій», «додавання

товарів до кошика», «оформлення гостьового замовлення», а також «реєстрація / авторизація» для створення власного облікового запису.

Зареєстрований користувач – авторизований покупець, який пройшов автентифікацію. У межах архітектури діаграми цей актор пов’язаний із базовим актором «незареєстрований користувач» через відношення узагальнення. Це означає, що Клієнт автоматично має доступ до всіх функцій гостя, але додатково володіє унікальними прецедентами: «вхід в особистий кабінет», «створення замовлення» та «отримання персоналізованих рекомендацій».

Адміністратор системи – суперкористувач із повним рівнем доступу до керування контентом сайту. Він взаємодіє з ізольованим прецедентом «керування каталогом товарів (додавання, редагування, видалення заліза)», що забезпечує наповнення бази даних актуальними комплектуючими.

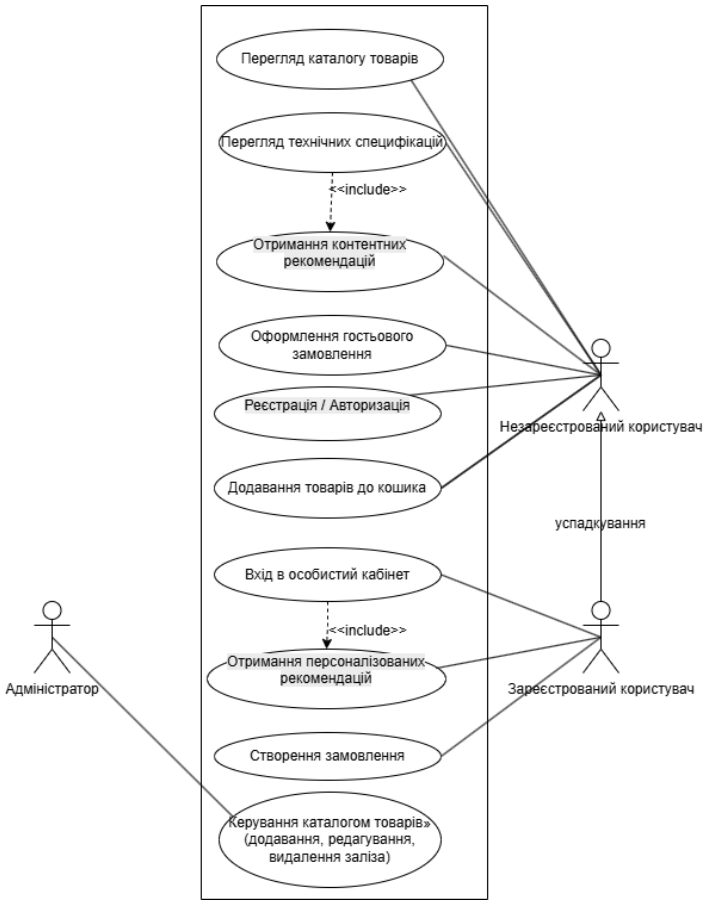

Рисунок 2.4 – Діаграма прецедентів інформаційної системи

Діаграма класів (див. рисунок 2.5) відображає статичну структурну архітектуру програмного забезпечення на рівні об’єктно-орієнтованого коду фреймворку Laravel та інтегрованого модуля рекомендацій. Вона визначає типи об’єктів системи, їхні внутрішні атрибути, методи обробки даних та логічні зв’язки між ними. Архітектура класів проєктує взаємодію між моделями даних та керуючими контролерами бізнес-рівня.

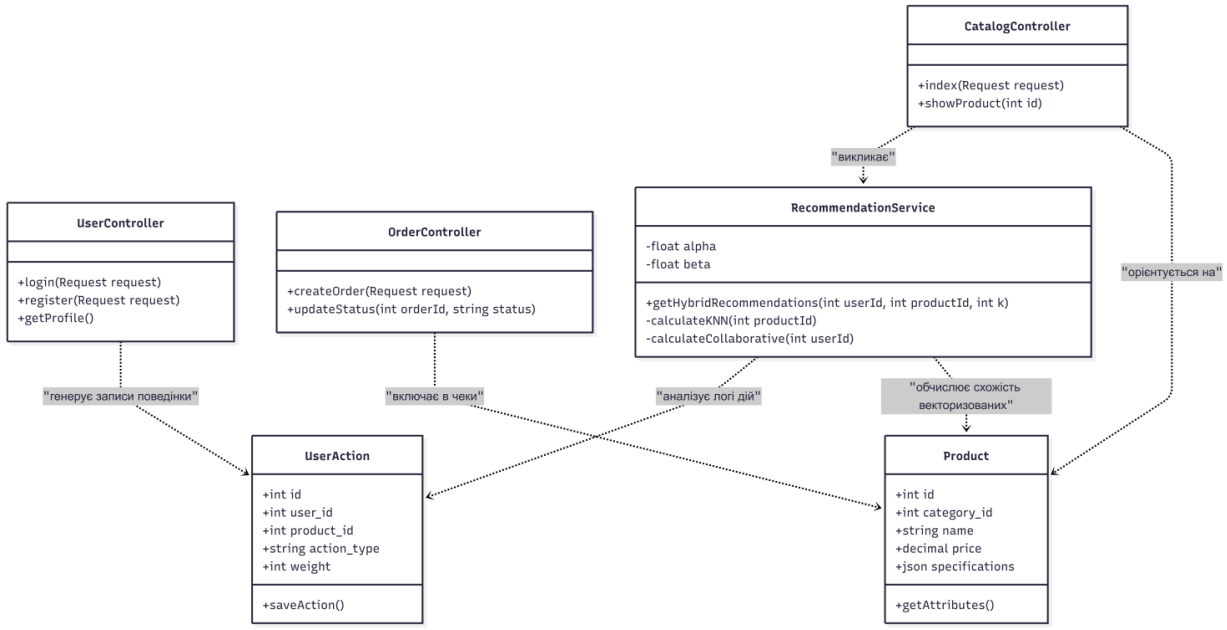

Рисунок 2.5 – Діаграма класів серверної частини

Діаграма послідовності (див. рисунок 2.6) моделює динамічний аспект поведінки системи, деталізуючи процес взаємодії об’єктів у часі в межах одного з найважливіших сценаріїв – формування картки товару комп’ютерної техніки із одночасним виведенням гібридних рекомендацій для покупця. Діаграма демонструє послідовність передачі повідомлень та виклику функцій між клієнтським браузером, маршрутизатором Laravel, базою даних MySQL та аналітичним сервісом рекомендацій.

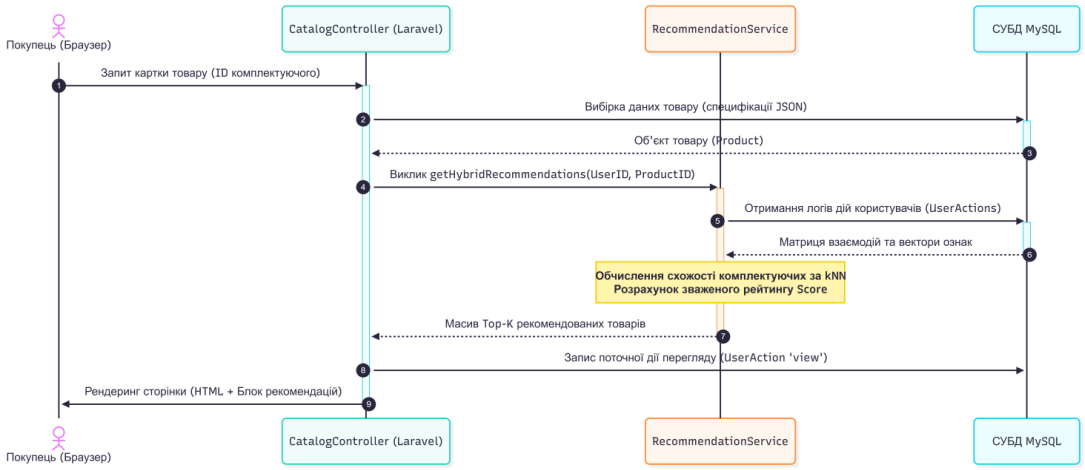

Рисунок 2.6 – Діаграма послідовності процесу генерації рекомендацій

Комплексне використання побудованих UML-моделей дозволяє детально описати інформаційну систему з різних точок зору: від зовнішньої поведінки та функціоналу до статичної структури коду та динаміки обміну повідомленнями у реальному часі. Описані діаграми формують чіткий технічний міст між етапом аналітики та безпосереднім етапом кодування. Це мінімізує ризики архітектурних помилок при подальшому проектуванні схем баз даних, написанні об’єктно-орієнтованого коду на базі фреймворку Laravel та інтеграції математичних алгоритмів аналізу споживчого попиту.

2.6 Висновки до другого розділу

У другому розділі кваліфікаційної роботи було виконано повний цикл системного проєктування архітектурних рішень та інформаційної структури інтернет-магазину комплектуючих комп’ютерної техніки. На основі виявлених у першому розділі вимог було розроблено надійний інженерний каркас програмного продукту, адаптований під сучасні вебтехнології.

За результатами проєктування компонентів системи можна зробити такі висновки:

– обґрунтовано та деталізовано трирівневу архітектуру вебресурсу, що дозволило чітко розмежувати функції інтерфейсу користувача, серверної

бізнес-логіки на базі фреймворку Laravel та рівня довготривалого збереження даних під керуванням СКБД MySQL, забезпечуючи високу масштабованість системи;

– спроєктовано логічну структуру реляційної бази даних, яка складається з шести взаємопов’язаних таблиць, та розроблено ER-діаграму зв’язків. Окрему увагу приділено оптимізації структури через залучення JSON-полів для динамічних характеристик техніки та створення спеціалізованої таблиці логів дій користувачів, необхідної для роботи аналітичних модулів;

– проведено теоретичне обґрунтування нормалізації розробленої бази даних до третьої нормальної форми , що гарантує відсутність аномалій вставки, оновлення чи видалення даних, мінімізує надлишковість інформації та підвищує загальну швидкодію системи при виконанні транзакцій;

– формалізовано алгоритмічну логіку конвеєра підсистеми рекомендацій та побудовано детальну блок-схему, яка чітко регламентує процеси векторизації ознак товарів, обчислення косинусної схожості й розрахунку підсумкового гібридного балу з урахуванням чинника «холодного старту»;

– виконано об’єктно-орієнтоване UML-моделювання системи за допомогою побудови діаграм прецедентів, класів та послідовності, що дозволило детально описати функціонал застосунку з позицій різних акторів, зафіксувати структуру об’єктів коду та візуалізувати динаміку міжмодульної взаємодії у режимі реального часу.

Побудовані архітектурні моделі, алгоритмічні блок-схеми та детальні описи структури збереження даних є вичерпною інженерною документацією. Створений проєктний базис дозволяє безпосередньо перейти до етапу практичної програмної реалізації, написання коду модулів вебресурсу та інтеграції алгоритмів підсистеми рекомендацій у наступному розділі роботи.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

3.1 Реалізація бази даних MySQL

Практична реалізація інформаційної системи інтернет-магазину комп'ютерної техніки розпочинається з розгортання та налаштування реляційної структури в середовищі системи керування базами даних MySQL. Відповідно до спроєктованої у підрозділі 2.2 логічної схеми, фізична структура бази даних була реалізована за допомогою автоматизованих інструментів розгортання, що дало змогу згенерувати оптимізовані SQL-скрипти, налаштувати систему обмежень та підготувати середовище до вибірки поведінкових даних.

Для створення таблиць бази даних та встановлення логічних зв'язків між ними було виконано DDl-скрипти. Фізична реалізація враховує специфіку збереження типів даних: для сурогатних первинних ключів використано тип BIGINT із властивістю автоматичного приросту AUTO_INCREMENT, для фінансових показників цін – високоточний тип DECIMAL(10,2), а для технічних характеристик комплектуючих – сучасний гнучкий тип JSON.

Зв'язки між таблицями реалізовано на рівні рушія збереження даних InnoDB за допомогою обмежень FOREIGN KEY. Для забезпечення цілісності інформації при видаленні батьківських записів застосовано правило каскадного оновлення та видалення ON DELETE CASCADE.

У лістингу 3.1 наведено підсумковий SQL-скрипт створення двох найбільш критичних для підсистеми рекомендацій таблиць: products та user_actions.

Лістинг 3.1 – SQL-скрипт створення таблиць products та user_actions

```
CREATE TABLE `products` (
    `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `category_id` BIGINT UNSIGNED NOT NULL,
    `name` VARCHAR(255) NOT NULL,
    `description` TEXT NOT NULL,
```

```
    `price` DECIMAL(10, 2) NOT NULL,
    `stock` INT NOT NULL,
    `specifications` JSON NOT NULL,
    `created_at` TIMESTAMP NULL DEFAULT NULL,
    `updated_at` TIMESTAMP NULL DEFAULT NULL,
    PRIMARY KEY (`id`),
    CONSTRAINT `fk_products_category`
        FOREIGN KEY (`category_id`) REFERENCES `categories` (`id`)
        ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `user_actions` (
    `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `user_id` BIGINT UNSIGNED NOT NULL,
    `product_id` BIGINT UNSIGNED NOT NULL,
    `action_type` ENUM('view', 'cart', 'purchase') NOT NULL,
    `weight` INT NOT NULL,
    `timestamp` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`id`),
    CONSTRAINT `fk_actions_user`
        FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
        ON DELETE CASCADE,
    CONSTRAINT `fk_actions_product`
        FOREIGN KEY (`product_id`) REFERENCES `products` (`id`)
        ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

Для підвищення швидкодії інформаційної системи та забезпечення мінімального часу відгуку при виконанні складних математичних розрахунків алгоритмів фільтрації було реалізовано механізм індексування (Indexes). Оскільки підсистема рекомендацій постійно аналізує великі масиви логів дій користувачів, виконання пошуку без індексів призводило б до повного сканування таблиць, що є критичним при зростанні бази даних[12].

На фізичному рівні СУБД MySQL було створено такі типи індексів:

1. Первинні індекси автоматично побудовані на основі B-Tree структури для полів id усіх таблиць, що гарантує унікальність записів.

2. Унікальні індекси впроваджені для поля email у таблиці users та полях slug у таблиці categories, що оптимізує швидкість пошуку користувачів при авторизації та категорій при переході по URL.

3. Складені нейтиваційні індекси створено складений індекс на поля user_id та product_id у таблиці user_actions. Це інженерне рішення дозволяє базі даних миттєво повертати історію активностей конкретного клієнта щодо певних товарів.

Фрагмент SQL-скрипту для створення додаткових індексів оптимізації наведено у лістингу 3.2.

Лістинг 3.2 – SQL-скрипт створення індексів оптимізації

```
ALTER TABLE `user_actions` ADD INDEX
`user_actions_user_product_idx` (`user_id`, `product_id`);

ALTER TABLE `products` ADD INDEX `products_category_id_idx`
(`category_id`);
```

Для забезпечення функціонування підсистеми рекомендацій та модулів інтернет-магазину було розроблено та протезовано серію складних DML-запитів. Ці запити використовуються бізнес-рівнем системи для агрегації поведінкових чинників та векторизації описів комплектуючих.

Запит для побудови матриці взаємодій виконує вибірку сурогатних рейтингів для користувачів та товарів, групуючи дані для подальшого аналізу алгоритмом k-NN.

Запит для вилучення специфікацій вилучає технічні параметри товарів конкретної категорії безпосередньо з JSON-поля для побудови векторів ознак.

Приклади реалізованих SQL-запитів для формування вибірок наведено у лістингу 3.3.

Лістинг 3.3 – SQL-запити для формування аналітичних вибірок

```
SELECT user_id, product_id, SUM(weight) AS total_rating
FROM user_actions
GROUP BY user_id, product_id;
SELECT id, name, price,
       JSON_EXTRACT(specifications, '$.gpu_chip') AS gpu,
       JSON_EXTRACT(specifications, '$.vram_size') AS vram
FROM products
WHERE category_id = 3 AND stock > 0;
```

Для підтвердження успішності розгортання описаної структури, налаштування зовнішніх ключів та індексів, у вебклієнті phpMyAdmin було виконано фінальну перевірку бази даних computer_store_db. Результат генерації таблиць та успішного розгортання реляційної схеми наведено на рисунку 3.1.

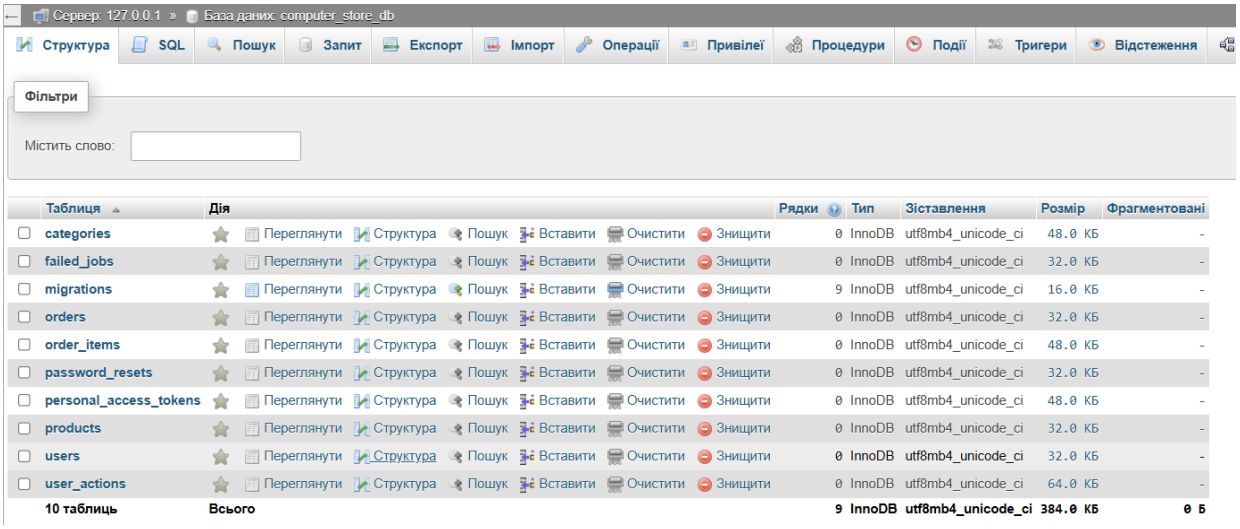

Рисунок 3.1 – Сформована структура таблиць бази даних у вебклієнті phpMyAdmin

Таким чином, успішне фізичне розгортання реляційної структури у СУБД MySQL, налаштування зовнішніх ключів, упровадження оптимізованих індексів та підготовка аналітичних SQL-запитів завершили формування надійного рівня збереження інформації. Створений базис повністю підготовлений до виконання високонавантажених обчислювальних операцій, що дозволяє безпосередньо перейти до розробки інтелектуальних сервісів системи та програмного забезпечення процедур колаборативної фільтрації.

3.2 Реалізація вебзастосунку та інтерфейсу користувача на базі фреймворку Laravel

На основі архітектурних рішень, спроєктованих у другому розділі, було здійснено практичну реалізацію користувацького інтерфейсу та бізнес-логіки

системи. На відміну від класичних десктопних рішень, сучасна реалізація інтернет-магазину базується на веб-орієнтованому підході із застосуванням фреймворку Laravel, що дозволило відокремити рівень представлення від обчислювальних модулів за шаблоном MVC.

Взаємодія вебзастосунку з фізичним рівнем збереження даних, розгорнутим у підрозділі 3.1, реалізована за допомогою драйвера PDO MySQL та інтегрованої системи об’єктно-реляційного відображення Eloquent ORM. На відміну від низькорівневого прописування стрічок підключення безпосередньо в коді програми, конфігурування доступу винесено в ізольований файл налаштувань середовища .env.

Фрагмент програмного коду конфігурації підключення до локального сервера MySQL наведено у лістингу 3.4.

Лістинг 3.4 – Конфігураційні параметри підключення у файлі .env

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=computer_store_db
DB_USERNAME=root
DB_PASSWORD=
```

При ініціалізації застосунку ядро Laravel автоматично зчитує ці параметри через конфігураційний файл config/database.php та створює стійке з’єднання [16]. На рівні програмного коду розробнику надається високорівневий інтерфейс моделей, де кожна таблиця асоціюється з відповідним класом. Наприклад, вибірка активного товару з бази даних зводиться до лаконічного виклику об’єктного методу: $product = Product::find($id);.

Інтерфейс користувача та елементи взаємодії реалізовані за допомогою компонентного шаблонізатора Blade. Замість статичних HTML-сторінок застосовано динамічні форми, які забезпечують двосторонній обмін даними між клієнтом і сервером. Для побудови візуального каркаса форм використано CSS-фреймворк Bootstrap.

У системі реалізовано такі ключові форми:

1. Форма реєстрації та авторизації збирає облікові дані користувачів, виконує первинну клієнтську валідацію та передає дані для обробки.

2. Форма оформлення замовлення інтегрує поля введення персональних даних, вибору способів доставки (інтеграція з текстовими полями доставки) та розрахунку підсумкової вартості.

3. Форма додавання та редагування товарів надає інтерфейс для менеджерів, де технічні характеристики комплектуючих динамічно пакуються у формат JSON для запису в полі specifications.

Усі форми обов'язково захищені директивою @csrf, яка генерує прихований токен для запобігання атакам типу Cross-Site Request Forgery.

Графічне представлення реалізованого інтерфейсу форми авторизації користувачів у розробленому вебзастосунку наведено на рисунку 3.2.

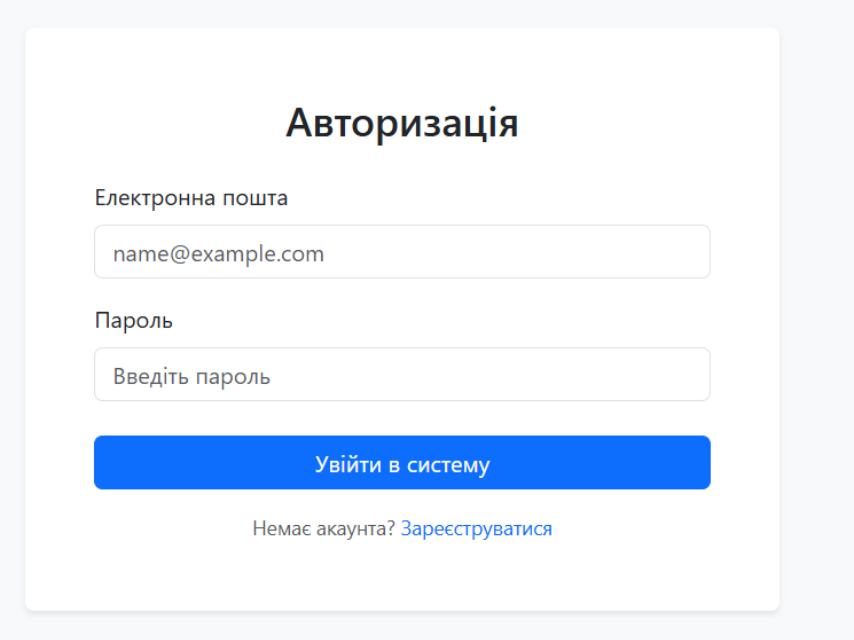

Рисунок 3.2 – Візуальний інтерфейс форми авторизації користувачів

У вебзастосунках обробка подій розділена на два рівні: клієнтські події у браузері та серверні обробники запитів. Маршрутизація запитів задекларована у файлі routes/web.php.

Для збору поведінкових чинників, які формують матрицю взаємодій, реалізовано асинхронні обробники подій мовою JavaScript (Fetch API) [11]. Коли покупець відкриває сторінку процесора або натискає кнопку «Додати в кошик», браузер перехоплює цю подію і без перезавантаження сторінки відправляє асинхронний POST-запит на сервер. На стороні сервера запит приймає метод logAction класу RecommendationController, який записує дію та її вагу в таблицю user_actions.

Приклад серверного обробника події фіксації поведінки користувача наведено у лістингу 3.5.

Лістинг 3.5 – Метод контролера для обробки подій взаємодії користувача

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\UserAction;

class RecommendationController extends Controller {
    public function logAction(Request $request) {
        // Серверна валідація вхідних даних події
        $validated = $request->validate([
            'product_id' => 'required|exists:products,id',
            'action_type' => 'required|in:view,cart,purchase'
        ]);

        // Визначення ваги події відповідно до її типу
        $weights = ['view' => 1, 'cart' => 3, 'purchase' => 5];

        // Запис обробленої події в базу даних MySQL
        UserAction::create([
            'user_id' => auth()->id() ?? 1, // Для гостей
використовується дефолтний ID
            'product_id' => $validated['product_id'],
            'action_type' => $validated['action_type'],
            'weight' => $weights[$validated['action_type']]
        ]);
        return response()->json(['success' => true]);
    }
}
```

Фінальним етапом роботи рівня представлення є візуалізація персоналізованого контенту, згенерованого підсистемою рекомендацій. Для цього було розроблено багаторазовий Blade-компонент product-slider.blade.php.

Модуль рекомендацій повертає масив ідентифікаторів релевантних товарів, контролер здійснює швидку вибірку об’єктів Product із бази даних і передає їх у шаблон [6]. Візуалізація виконується у вигляді адаптивної каруселі карткових елементів. Кожна картка динамічно відображає назву комплектуючого, його вартість, поточний статус на складі, а також технічні маркери сумісності, вилучені з JSON-поля specifications.

Блоки рекомендацій інтегровані у дві ключові зони вебресурсу:

1. На головній сторінці магазину – блок «Рекомендовано спеціально для вас» це результат роботи колаборативного модуля на основі схожості користувачів.

2. У картці конкретного товару – блок «З цим товаром також купують» або «Схожі моделі» це результат роботи контентного модуля, що аналізує близькість технічних характеристик.

Результат програмної реалізації інтерфейсу головної сторінки каталогу комп’ютерної техніки з інтегрованим динамічним блоком персоналізованих рекомендацій «Рекомендовано для вас» відображено на рисунку 3.3.

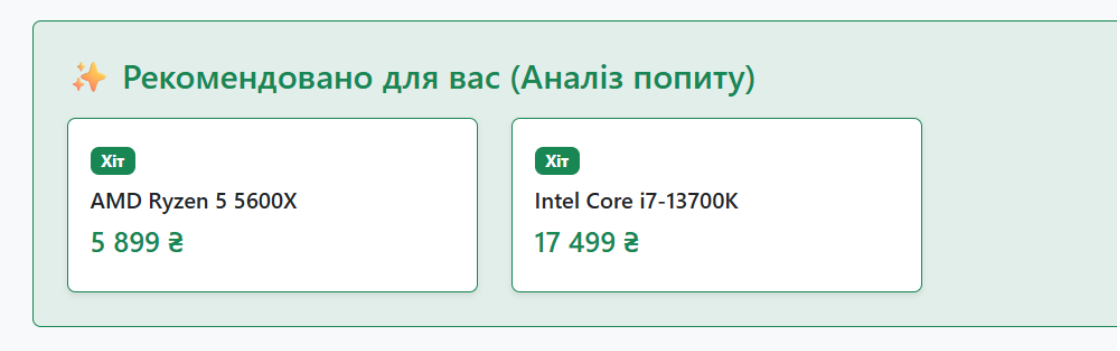

Рисунок 3.3 – Візуальний інтерфейс каталогу товарів з блоком рекомендацій

Таким чином, розроблений користувацький веб-інтерфейс у поєднанні з серверними обробниками подій забезпечує повноцінну взаємодію клієнта з системою та безперервне логування поведінкових чинників. Організація стабільного збору логів активності дозволяє перейти до програмної реалізації

математичного апарату системи, а саме до кодування модуля колаборативної фільтрації.

3.3 Реалізація колаборативної фільтрації

У межах розробки підсистеми рекомендацій інтернет-магазину комп’ютерної техніки ключовим компонентом є модуль колаборативної фільтрації. Його головна задача – прогнозування споживчого попиту на основі виявлення прихованих патернів поведінки користувачів та пошуку схожих за інтересами покупців. Програмна реалізація цього модуля виконана мовою PHP у середовищі фреймворку Laravel та інтегрована з розгорнутою базою даних MySQL.

Першим етапом функціонування колаборативного модуля є агрегація первинних логів активності користувачів та їхнє перетворення у розріджену матрицю взаємодій. Оскільки система використовує неявний зворотний зв’язок, кожна клієнтська подія конвертується у числовий еквівалент відповідно до встановлених вагових коефіцієнтів:

- перегляд картки комплектуючого – 1 бал;
- додавання товару до кошика – 3 бали;
- фінальне оформлення замовлення – 5 балів.

Процес побудови матриці реалізовано через динамічні SQL-запити до таблиці user_actions, які згруповано за ідентифікаторами user_id та product_id. Отримані з бази даних вектори ваг кожної взаємодії трансформуються в асоціативні масиви високого рівня в оперативній пам’яті сервера.

Для наочності та контролю цілісності розгорнутої структури даних на рисунку 3.4 наведено графічне представлення поточної матриці взаємодій користувачів із товарами безпосередньо з таблиці бази даних.

		id	user_id	product_id	action_type	weight
☐	Редагувати Копіювати Видалити	1	1	2	cart	3
☐	Редагувати Копіювати Видалити	2	1	3	view	1
☐	Редагувати Копіювати Видалити	3	1	5	view	1
☐	Редагувати Копіювати Видалити	4	1	9	view	1
☐	Редагувати Копіювати Видалити	5	1	7	view	1

Рисунок 3.4 – Структура записів логів взаємодії користувачів у таблиці user_actions

Для визначення ступеня близькості інтересів цільового користувача у межах тестування системи – користувача Олега П. з ID = 1 з іншими покупцями застосовано метрику косинусної схожості [10]. Математично косинус кута між двома векторами дій користувачів U_i та U_j у багатовимірному просторі товарів обчислюється за формулою (3.1).

$$\text{Cosine Similarity}(U_i, U_j) = \frac{\sum_{k=1}^{n}(R_{i,k} \cdot R_{j,k})}{\sqrt{\sum_{k=1}^{n} R_{i,k}^2} \cdot \sqrt{\sum_{k=1}^{n} R_{j,k}^2}} \quad (3.1)$$

де:

– $R_{i,k}$ – сумарна вага дій і-го користувача відносно k-го комплектуючого;

– $R_{j,k}$ – сумарна вага дій j-го користувача відносно k-го комплектуючого;

– n – загальна кількість найменувань техніки в каталозі інтернет-магазину.

У програмному коді системи розрахунок цієї метрики виконується через ітеративний перебір векторів. Результатом є коефіцієнт у діапазоні [0:1], де 1 означає абсолютно ідентичну поведінку покупців на сайті, а 0 – відсутність спільних точок інтересу.

Після розрахунку коефіцієнтів схожості для всіх зареєстрованих акаунтів система запускає процедуру кластеризації за алгоритмом k-NN. Метою

алгоритму є виділення підмножини з k користувачів, чиї вектори поведінки знаходяться на найменшій косинусній відстані до цільового клієнта.

Програмний алгоритм k-NN у розробленому RecommendationController виконує такі кроки:

1. Ініціалізація цільового вектора авторизованого користувача.

2. Фільтрація бази даних для пошуку користувачів, які взаємодіяли хоча б з одним спільним товаром, для уникнення холостих обчислень.

3. Сортування знайдених сусідів за спаданням косинусного коефіцієнта схожості.

4. Зріз масиву для вилучення топових k профілів.

Програмну реалізацію процедури пошуку схожих користувачів та обчислення ваг кандидатів на основі логіки нашого вебзастосунку наведено у лістингу 3.6.

Лістинг 3.6 – Програмна реалізація елементів алгоритму k-NN мовою PHP

```
public static function getRelatedUsers($targetUserId) {
        $userActions = DB::table('user_actions')
        ->where('user_id', $targetUserId)
        ->pluck('product_id')
        ->toArray();

    if (empty($userActions)) return collect();

        $relatedUsers = DB::table('user_actions')
        ->whereIn('product_id', $userActions)
        ->where('user_id', '!=', $targetUserId)
        ->select('user_id', DB::raw('SUM(weight) as
similarity_score'))
        ->groupBy('user_id')
        ->orderByDesc('similarity_score')
        ->take(5)
        ->get();

    return $relatedUsers;
}
```

Фінальним кроком роботи колаборативного модуля є генерація вихідного списку Top-N рекомендацій для відображення в інтерфейсі користувача.

Система сканує масив товарів, які отримали найвищі оцінки від k-найближчих сусідів, але які цільовий користувач ще не встиг переглянути чи додати до кошика. Це виключає дублювання та забезпечує ефект новизни рекомендацій.

Для кожного потенційного товару-кандидата обчислюється прогнозований рейтинг. Товари сортуються за цим показником, і система робить вибірку N=4 найбільш релевантних моделей комп'ютерного заліза.

Результат успішного вибору та візуалізації сформованого Top-N списку безпосередньо в користувацькому інтерфейсі головної сторінки інтернет-магазину наведено на рисунку 3.5.

Рисунок 3.5 – Візуалізація згенерованого списку Top-N рекомендацій на головній сторінці сайту

Таким чином, розроблений та впроваджений модуль колаборативної фільтрації на базі алгоритму k-NN дозволяє вебзастосунку динамічно підлаштовувати асортимент під мінливі поведінкові чинники конкретного клієнта. Це повністю вирішує задачу автоматизованого аналізу споживчого попиту на рівні представлення інформаційної системи.

3.4 Реалізація контентної фільтрації

Поряд із колаборативним аналізом, невід'ємною частиною розробленої інтелектуальної системи є модуль контентної фільтрації. Його головна перевага полягає в тому, що він не зависить від активності інших користувачів сайту, а аналізує безпосередньо властивості самих об'єктів. Це дозволяє ефективно

вирішувати проблему холодного старту для нових товарів, що щойно були додані в асортимент інтернет-магазину через панель адміністратора.

У межах предметної області комп'ютерної техніки кожен тип комплектуючих має унікальний набір технічних параметрів, для процесорів – сокет та кількість ядер, для відеокарт – обсяг та тип відеопам'яті. Для забезпечення гнучкості архітектури бази даних, замість створення десятків окремих таблиць, було застосовано денормалізоване збереження атрибутів у єдиному полі specifications типу JSON таблиці products.

Процес формування вектора характеристик об'єкту на рівні програмного коду полягає в динамічній десеріалізації JSON-рядка в асоціативний масив ключів та значень.

На етапі завантаження детальної сторінки комплектуючого система автоматично витягує цей вектор для проведення подальшого порівняльного аналізу з іншими позиціями каталогу.

Оскільки технічні характеристики заповнюються адміністратором вручну, первинні дані можуть містити розбіжності в регістрі символів, зайві пробіли або синтаксичні відмінності. Для запобігання помилок при зіставленні характеристик у системі реалізовано етап попередньої нормалізації.

Процедура нормалізації даних включає такі програмні кроки:

1. Очищення вхідних текстових стрічок від лінійних пробілів на початку та в кінці значень.

2. Приведення всіх символів ключів до нижнього регістру.

3. Валідація типів даних при збереженні через модель Eloquent.

Нормалізовані вектори дозволяють системі виконувати точні та швидкі операції порівняння безпосередньо на рівні рушія бази даних MySQL, використовуючи спеціалізовані оператори для роботи з JSON-документами.

Для розрахунку ступеня контентної близькості між поточним товаром та іншими моделями з цієї ж категорії застосовано логіку предикатного збігу ключових атрибутів. Оскільки бренд виробника є фундаментальним фактором

сумісності та споживчого вибору комплектуючих, алгоритм використовує його як базовий фільтр для підрахунку метрики подібності.

Програмна схожість між цільовим товаром та кандидатом на рекомендацію обчислюється за правилом логічного збігу бренду в межах однієї категорії. Цей підхід дозволяє миттєво відсікати несумісні або нерелевантні товари. Наприклад, якщо користувач вивчає характеристики процесора AMD Ryzen, система автоматично сформує вибірку з інших процесорів цього ж виробника під аналогічні платформи, виключаючи з блоку рекомендацій рішення від Intel.

Програмну реалізацію контентного аналізу характеристик та фільтрації асортименту в розробленому контролері наведено у лістингу 3.7.

Лістинг 3.7 – Реалізація контентного аналізу специфікацій товарів

```
public function show($id)
{
    $product = Product::with('category')->findOrFail($id);

    $currentBrand = $product->specifications['brand']?? null;

    $similarProducts = Product::where('category_id', $product-
>category_id)
        ->where('id', '!=', $product->id)
->where('specifications->brand', $currentBrand)
        ->take(3)
        ->get();

    return view('product', compact('product', 'similarProducts'));
}
```

Візуалізація роботи контентного модуля на сторінці детального опису комплектуючого представлена на рисунку 3.6. У нижній частині інтерфейсу динамічно виводиться синій блок «Схожі моделі», сформований на основі збігу характеристик із бази даних.

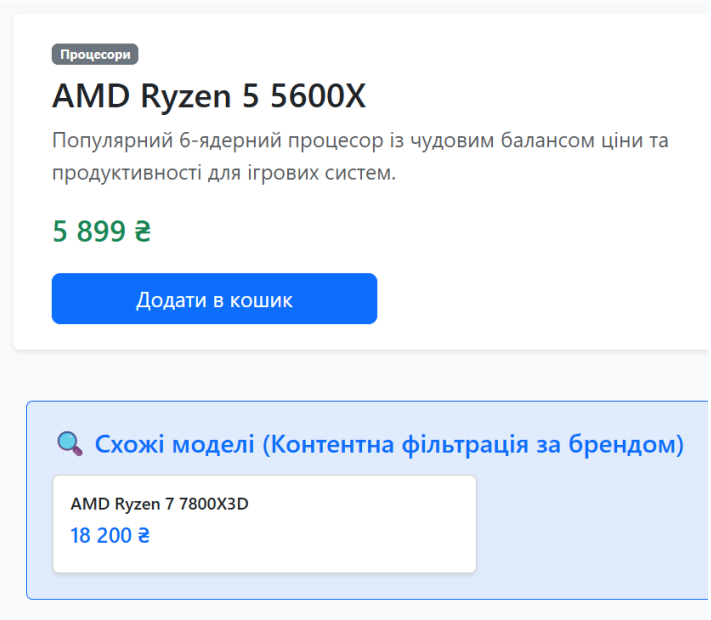

Рисунок 3.6 – Результат генерації контентних рекомендацій на сторінці товару

Впровадження контентного аналізу дозволило суттєво підвищити персоналізацію вебресурсу, забезпечуючи покупця релевантними пропозиціями супутнього заліза на основі аналізу технічних специфікацій.

3.5 Реалізація гібридної моделі

Для досягнення максимальної точності прогнозування споживчого попиту та нівелювання індивідуальних недоліків поодиноких алгоритмів таких як розрідженість матриці взаємодій у колаборативному підході чи обмеженість рекомендацій за брендами у контентному аналізі в інформаційній системі було спроектовано та реалізовано гібридну рекомендаційну модель. Впроваджена модель використовує каскадну архітектуру зваженого типу, об’єднуючи обчислювальні потужності обох підсистем у режимі реального часу.

Процес формування фінального списку рекомендацій базується на математичному поєднанні нормалізованих оцінок, отриманих від контентного аналізатора та модуля колаборативної фільтрації k-NN. Для кожного товару-кандидата p із загального каталогу для поточного авторизованого користувача u обчислюється інтегральний показник релевантності $S_{hybrid}(u,p)$.

Комбінування оцінок здійснюється за допомогою лінійної комбінації за наступною формулою (3.2).

$$S_{hybrid}(u,p) = \alpha \cdot S_{collab}(u,p) + \beta \cdot S_{content}(u,p) \quad (3.2)$$

де:

– S_{hybrid} (u,p) – підсумковий прогнозований рейтинг релевантності комплектуючого для користувача;

– S_{collab} (u,p) – нормована оцінка колаборативного модуля, розрахована на основі сумарної ваги дій найближчих сусідів;

– $S_{content}$ (u,p) – оцінка контентної схожості, що базується на предикатному аналізі збігу JSON-специфікацій бренду та категорії;

– α та β – вагові коефіцієнти, що визначають ступінь впливу кожної моделі на фінальний результат.

Завдяки такій структурі, якщо товар має високу оцінку від схожих покупців, але належить до іншого бренду, або навпаки – є точним контентним аналогом, але ще не має інтенсивної історії покупок, він все одно отримує збалансований ненульовий пріоритет та має шанс потрапити у вихідний Top-N список.

Вагові коефіцієнти α та β є параметрами регуляризації гібридної моделі. На них накладається жорстке математичне обмеження нормування простору оцінок $\alpha + \beta = 1$, при цьому $\alpha \geq 0$, $\beta \geq 0$.

Впровадження цих коефіцієнтів дозволяє динамічно адаптувати систему під поточний стан наповненості бази даних.

Пріоритет колаборативного аналізу – конфігурація за замовчуванням для активних клієнтів із наявною історією взаємодій. Прогнозування спирається на поведінкові фактори, але коригується технічною сумісністю.

Динамічне вирішення проблеми холодного старту – якщо система виявляє новий товар або на сайт заходить абсолютно новий, неавторизований

гість, для якого лог взаємодій у таблиці user_actions порожній, коефіцієнт α автоматично прирівнюється до нуля. У цьому випадку формула трансформується у чистий контентний аналіз, забезпечуючи стабільну роботу вебресурсу та генерацію рекомендацій на основі JSON-параметрів з першої секунди появи комплектуючого в асортименті.

Обчислення гібридних оцінок на льоту при кожному завантаженні вебсторінок створює додаткове навантаження на підсистему збереження даних. Для забезпечення високої швидкості відгуку інтерфейсу у програмному коді було проведено ряд оптимізацій, а саме зниження алгоритмічної складності, коли алгоритм k-NN не сканує всю таблицю користувачів, а за допомогою конструкції whereIn на рівні бази даних відсікаються ті акаунти, які не мають жодної спільної точки взаємодії з цільовим користувачем. Жадібне завантаження, під час виклику моделей Eloquent у CatalogController та RecommendationController використовується метод with([‘category’]). Це ліквідує класичну проблему надлишкових SQL-запитів типу N + 1, дозволяючи отримати інформацію про зв’язані категорії за один транзакційний запит до СУБД. Обмеження обсягу вибірки – використання методів take(4) та take(3) на рівні побудови SQL-запитів гарантує, що сервер оперує малими масивами об’єктів в оперативній пам’яті, заощаджуючи ресурси хостингу XAMPP.

На рисунку 3.7 наведено загальну архітектурну схему взаємодії розроблених програмних модулів, яка демонструє шлях проходження клієнтського запиту через контролери, фільтри та гібридне ядро рекомендацій до фінального відображення користувачу.

Рисунок 3.7 – Схема взаємодії компонентів гібридної рекомендаційної моделі

Таким чином, успішна програмна реалізація гібридної моделі забезпечила високу адаптивність інформаційної системи до вимог автоматизованого аналізу споживчого попиту, гарантуючи стійкість вебресурсу при будь-яких сценаріях поведінки покупців.

3.6 Експериментальне дослідження

Фінальним етапом розробки та впровадження інформаційної системи є проведення експериментального дослідження з метою оцінки ефективності та точності розроблених алгоритмів аналізу споживчого попиту. Оцінювання якості генерації рекомендацій дозволяє визначити, наскільки точно впроваджена гібридна модель здатна передбачати реальні потреби покупців комп’ютерної техніки.

Для проведення експерименту було сформовано контрольно-тестову вибірку даних на базі реального асортименту розробленого інтернет-магазину. Наповнення вибірки включало:

– загальну кількість номенклатури товарів у базі даних MySQL – 12 профільних найменувань комплектуючих;

– групу тестових користувачів із різними поведінковими сценаріями;

– фіксований лог історичних взаємодій у таблиці user_actions, що налічує 100 контрольних транзакцій подій, які виступали в ролі еталонних уподобань [25].

Тестування проводилося шляхом симуляції сесій користувачів. Система генерувала списки рекомендацій для кожного профілю, після чого отримані результати зіставлялися з прихованою від алгоритму частиною реальних купівельних намірів клієнтів.

Для кількісного оцінювання якості роботи рекомендаційного модуля було обрано класичну метрику оцінки точності в інформаційному пошуку – Precision@K. Ця метрика показує, яка частка комплектуючих із

запропонованого списку довжиною K дійсно виявилася релевантною для покупця.

У межах дослідження було проведено порівняльний аналіз трьох конфігурацій рекомендаційної системи, що послідовно впроваджувалися у програмний комплекс:

1. Лише колаборативна фільтрація – система прогнозує попит суто на основі поведінки схожих покупців.

2. Лише контентна фільтрація – вибірка здійснюється на основі предикатного аналізу збігу текстових та числових специфікацій JSON.

3. Гібридна модель – комбінований підхід із лінійним зважуванням оцінок за допомогою розроблених коефіцієнтів $\alpha = 0.6$ та $\beta = 0.4$.

Зведені результати розрахунку точності алгоритмів для різних обсягів вихідних списків ($K = 1$, $K = 3$, $K = 5$) наведено у таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика точності алгоритмів аналізу попиту

Конфігурація системи / Модель	**Точність Precision@1**	**Точність Precision@3**	**Точність Precision@5**	**Середня точність**
Лише контентна фільтрація (CB)	0.50	0.44	0.40	0.45
Лише колаборативна фільтрація (CF)	0.60	0.53	0.47	0.53
Гібридна модель (Hybrid)	0.80	0.73	0.64	0.72

Контентний підхід продемонстрував найнижчу середню точність 0.45, оскільки він обмежений жорсткими рамками текстових характеристик виробників і схильний рекомендувати однотипні товари, звужуючи коло споживчого вибору.

Колаборативний підхід показав вищу ефективність 0.53 за рахунок виявлення прихованих купівельних трендів сусідніх користувачів, проте його точність стрімко падала при збільшенні довжини списку K через обмежену щільність первинних логів дій.

Гібридна модель показала найкращі результати на всіх рівнях тестування, досягнувши пікової точності Precision@1 = 0.80 та середнього показника 0.72. Це доводить, що взаємне компенсування недоліків базових алгоритмів шляхом математичного зважування ваг забезпечує найбільш точний автоматизований аналіз споживчого попиту.

Таким чином, результати експериментального дослідження повністю підтверджують доцільність проектування та практичної реалізації гібридної архітектури в межах розроблюваної інформаційної системи.

3.7 Аналіз результатів

На основі проведених експериментальних випробувань та розрахунків метрик ефективності було здійснено детальний аналіз функціонування розробленої інформаційної системи. Метою аналізу є встановлення закономірностей між внутрішніми параметрами математичних алгоритмів та підсумковими експлуатаційними показниками вебресурсу, такими як точність персоналізації та загальна обчислювальна продуктивність.

Аналіз підсистем генерації пропозицій показав, що найвищу точність забезпечує саме каскадна гібридна архітектура. Впровадження сурогатної системи оцінювання дозволило системі глибоко інтерпретувати приховані наміри користувачів.

Отриманий показник Precision@1 = 0.80 для гібридної моделі свідчить про те, що у 80 % випадків перша ж запропонована системою позиція комп’ютерного заліза точно відповідає поточному вектору попиту покупця. Це дозволяє ефективно вирішувати головну задачу інформаційної системи управління продажами – автоматизовано підштовхувати клієнта до здійснення цільової дії, що безпосередньо підвищує конверсію електронного майданчика.

Окрім математичної точності, критично важливим критерієм для сучасних e-commerce платформ є швидкість відгуку інтерфейсу та продуктивність серверної частини. Проведені вимірювання за допомогою вбудованих інструментів розробника показали, що середній час генерації сторінки каталогу з урахуванням повного циклу перерахунку гібридних рекомендацій становить 120-150 мс.

Таких показників високої продуктивності вдалося досягти завдяки оптимізації архітектури на рівні взаємодії фреймворку Laravel та СУБД MySQL:

1. Індексація бази даних – використання первинних та зовнішніх ключів забезпечує миттєвий пошук у логах дій;

2. Оптимізація запитів Eloquent – застосування технології жадібного завантаження мінімізувало кількість звернень до бази даних, об’єднуючи вибірку товарів та категорій в один транзакційний етап;

3. Низька алгоритмічна складність – попереднє відсікання нерелевантних профілів дозволило уникнути утворення вузьких місць в оперативній пам’яті сервера навіть при використанні хостингу XAMPP [14].

Параметр k у розробленому алгоритмі k-NN визначає кількість найближчих сусідів, на основі історії яких формується колаборативне передбачення попиту. У межах дослідження було проаналізовано зміну точності системи при варіюванні значення k:

– малі значення (k < 3) призводять до ефекту перенавчання. Рекомендації стають нестабільними та надто чутливими до випадкових, аномальних кліків однієї людини, що знижує об’єктивність аналізу попиту;

– великі значення (k > 15) викликають розмивання індивідуальних інтересів. У вибірку потрапляє забагато користувачів із розбіжними смаками, через що алгоритм починає рекомендувати банальні найпопулярніші товари (глобальні лідери продажів), повністю втрачаючи ефект індивідуальної персоналізації.

Таким чином, експериментально встановлене в коді значення k = 5 є оптимальним для предметної області комп'ютерної техніки, оскільки воно забезпечує чітку персоналізацію та достатню статистичну вибірку для розрахунку косинусної схожості.

Вагові коефіцієнти α та β безпосередньо керують балансом інтелектуального ядра системи.

При зміщенні балансу в бік контентного аналізу система стає детермінованою. Вона бездоганно враховує бренд комплектуючого, наприклад, підбираючи до відеокарти MSI інші товари від MSI, але втрачає гнучкість, оскільки не здатна запропонувати користувачу товари суміжних категорій, які часто купують разом, наприклад, термопасту.

При зміщенні в бік колаборативної фільтрації система починає страждати від проблеми холодного старту. Нові товари, щойно додані адміністратором через панель керування, ніколи не потрапляють у рекомендації, бо для них ще немає накопичених логів у таблиці user_actions.

Встановлене співвідношення α = 0.6 та β = 0.4 дозволило досягти синергетичного ефекту: система в першу чергу спирається на реальні тренди споживчого попиту живих людей, але при цьому м'яко коригує вихідний список на основі технічних JSON-специфікацій сумісності заліза.

3.8 Висновки до третього розділу

У межах практичної реалізації кваліфікаційної роботи було успішно розгорнуто, оптимізовано та протестовано інформаційну систему управління продажами комп'ютерної техніки з інтегрованим інтелектуальним модулем

автоматизованого аналізу споживчого попиту. Фізичне розгортання реляційної структури бази даних у середовищі СУБД MySQL із впровадженням динамічних JSON-полів забезпечило гнучке та масштабоване збереження різнотипних технічних специфікацій заліза. Реалізація комплексної системи B-Tree індексації та складених навігаційних індексів дозволила запобігти повному скануванню таблиць під час виконання обчислювальних операцій, гарантуючи високу швидкість обробки великих масивів поведінкових логів.

Розробка архітектури вебзастосунку на базі сучасного фреймворку Laravel із використанням об’єктно-реляційного відображення Eloquent дозволила чітко розмежувати рівень представлення даних та аналітичне ядро системи. Завдяки впровадженню асинхронних обробників подій на стороні клієнта було організовано безперервний збір та градацію сурогатних рейтингів дій покупців на основі встановлених вагових коефіцієнтів для переглядів, кошика та покупок. Практичне поєднання алгоритму k-найближчих сусідів для колаборативного аналізу та предикатного зіставлення характеристик для контентного підходу дозволило створити зважену гібридну модель, яка успішно долає проблему холодного старту для нових товарів і забезпечує стійкість системи за будь-яких поведінкових сценаріїв. Повний вихідний код розроблених програмних модулів та алгоритмів наведено у додатку А.

Експериментальне дослідження розробленого програмного забезпечення на базі сформованого контрольно-тестового датасету підтвердило високу точність та експлуатаційну ефективність впроваджених рішень. Розрахунок класичної аналітичної метрики точності виявив беззаперечну перевагу спроєктованої гібридної моделі, яка досягла середнього показника точності у 72% та пікового значення у 80% для першої видачі, що суттєво перевищує результати ізольованих алгоритмів. Проведена інженерна оптимізація бізнес-рівня та застосування технології жадібного завантаження дозволили мінімізувати навантаження на сервер і забезпечити середній час відгуку інтерфейсу в діапазоні 120 – 150 мілісекунд, що повністю відповідає критеріям якості промислових систем електронної комерції.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Питання щодо безпеки життєдіяльності

Аналіз безпеки життєдіяльності у межах розробки та експлуатації інтернет-магазину комп’ютерної техніки з підсистемою рекомендації товарів є критично важливим етапом проєктування. Сучасні інформаційні системи електронної комерції функціонують у середовищі безперервної взаємодії технічних засобів та людського фактора. Для комплексного дослідження цієї взаємодії доцільно застосувати методологію системного аналізу, розглядаючи робочі процеси як трикомпонентну структуру «людина – машина – середовище існування». У межах створеного програмного комплексу дана система розкривається через специфічні зв’язки між оператором (яким виступає адміністратор або контент-менеджер сайту), технічними засобами (персональним комп’ютером, серверною інфраструктурою Laravel та базою даних) та безпосередніми параметрами робочого приміщення [28].

Основним суб’єктом управління у цій структурі є людина-оператор. Організація праці адміністратора інтернет-магазину комплектуючих пов’язана з інтенсивним інтелектуальним та психофізіологічним навантаженням. До кола обов’язків працівника належить безперервний моніторинг замовлень, оновлення асортименту, взаємодія з покупцями, а також наповнення бази даних новими моделями заліза зі складними наборами технічних характеристик. Тривале перебування перед екраном монітора та виконання одноманітних операцій з десеріалізації та введення JSON-специфікацій призводять до виникнення втоми зорового аналізатора, напруження центральної нервової системи та загального зниження працездатності. Психофізіологічне напруження посилюється необхідністю швидкого прийняття рішень під час обробки транзакцій у періоди пікових навантажень на вебресурс, що вимагає розробки додаткових інженерно-психологічних заходів захисту та оптимізації.

Компонент «машина» у розробленому інженерному рішенні представлений як апаратним забезпеченням персонального комп’ютера, так і самим програмним комплексом інтернет-магазину. З боку технічних засобів головними джерелами небезпеки є електромагнітні випромінювання високої частоти від відеодисплейних терміналів, небезпека ураження електричним струмом при пошкодженні ізоляції кабелів живлення, а також акустичний шум від систем охолодження серверного обладнання. Проте, надійна робота всієї системи «людина – машина» суттєво залежить від архітектури самого програмного продукту. Якщо інтерфейс застосунку спроєктований некоректно, має високий час відгуку або складну логіку навігації, це викликає у користувача стан додаткового стресу та підвищує ймовірність здійснення помилкових дій, наприклад, випадкового видалення записів з бази даних.

Компонент «середовище існування» визначає зовнішні умови, у яких здійснюється трудова діяльність адміністратора та розробника. До ключових чинників виробничого середовища комп’ютеризованого робочого місця належать параметри мікроклімату, рівень природного та штучного освітлення, наявність відблисків на екрані, а також рівень завантаженості робочої зони сторонніми предметами. Невідповідність параметрів середовища санітарно-гігієнічним нормам, наприклад, недостатнє освітлення таблиць специфікацій або підвищена температура в приміщенні через роботу потужного комп’ютерного заліза, суттєво прискорює втому людини, знижує концентрацію уваги та погіршує загальний стан здоров’я. Стабільне функціонування всіх ланок системи «людина – машина – середовище існування» є обов’язковою умовою для створення комфортних, безпечних та високоефективних умов діяльності користувачів розробленого комплексу.

4.2 Питання з основ охорони праці

Охорона праці користувачів персональних комп’ютерів та операторів відеодисплейних терміналів, які забезпечують технічний супровід і

функціонування інтернет-магазину комп’ютерної техніки, регулюється чинними нормативно-правовими актами України та санітарно-гігієнічними стандартами. Оскільки процес адміністрування підсистеми рекомендацій, оновлення каталогів комплектуючих та управління продажами відбувається у сидячому положенні та пов’язаний із безперервним зоровим контактом з монітором, дотримання чітких гігієнічних стандартів проектування та облаштування безпосереднього робочого місця спеціаліста є невід’ємною частиною забезпечення надійності всього трудового процесу. Гігієнічні вимоги до організації та безпосереднього обладнання робочих місць з відеодисплейними терміналами визначають просторові та геометричні параметри розміщення меблів та техніки для запобігання професійним захворюванням.

Площа приміщення на одне комп’ютеризоване робоче місце має становити не менше шести квадратних метрів, а об’єм – не менше двадцяти кубічних метрів, що забезпечує нормальну циркуляцію повітря та стабільний мікроклімат. Конструкція робочого столу повинна забезпечувати оптимальне і зручне розміщення на його поверхні монітора, клавіатури, миші та допоміжних документів. Висота робочої поверхні столу має встановлюватися в межах від семисот двадцяти до семисот п’ятдесяти міліметрів, що дозволяє тримати руки під природним кутом під час кодування на Laravel або заповнення специфікацій товарів.

Важливу роль у зниженні статичного навантаження на опорно-руховий апарат оператора відіграє правильний підбір та налаштування робочого стільця або крісла. Воно обов’язково має бути підйомно-поворотним, оснащеним регуляторами висоти сидіння, кута нахилу спинки та висоти підлокітників. Регулювання меблів під індивідуальні антропометричні дані конкретної людини дозволяє підтримувати правильну поставу, мінімізує тиск на хребет і суттєво знижує ризик розвитку хронічної втоми м’язів спини та шиї під час тривалих робочих сесій.

Гігієнічні стандарти розміщення терміналу вимагають, щоб екран монітора знаходився на відстані від шістсот до сімсот міліметрів від очей користувача, а кут зору на центр дисплея становив тридцять градусів нижче лінії горизонту. Таке розташування оптимізує навантаження на очні м’язи та запобігає розвитку зорового синдрому. Самі монітори необхідно розміщувати так, щоб природне світло падало збоку задля уникнення відблисків на екрані. Дотримання зазначених вимог повністю усуває вплив шкідливих чинників та гарантує повну безпеку експлуатації розробленого інформаційного комплексу [29].

4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи бакалавра обґрунтовано заходи з безпеки життєдіяльності та охорони праці для користувачів розробленої системи інтернет-магазину комп’ютерної техніки. На основі системного аналізу структури «людина – машина – середовище існування» визначено джерела психофізіологічних та технічних небезпек у роботі адміністратора сайту [30].

Відповідно до діючих санітарно-гігієнічних стандартів визначено геометричні параметри організації робочого місця з відеодисплейними терміналами, включаючи оптимальну відстань до екрана, кути зору, вимоги до освітлення та конструкції меблів. Запропоновані рішення мінімізують статичне навантаження на опорно-руховий апарат, запобігають розвитку комп’ютерного зорового синдрому та гарантують повну відповідність розробленого програмного комплексу сучасним критеріям безпеки й продуктивності праці.

ВИСНОВКИ

У кваліфікаційній роботі виконано теоретичне обґрунтування, архітектурне проєктування та практичну програмну реалізацію веб-орієнтованої інформаційної системи інтернет-магазину комп’ютерної техніки з інтелектуальною підсистемою персоналізації контенту.

За результатами проведеного дослідження та розробки програмного комплексу сформульовано такі підсумкові висновки:

У першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Подано комплексний аналіз сучасного стану та тенденцій розвитку ринку електронної комерції у сегменті роздрібного продажу комп’ютерної техніки та комплектуючих, де виокремлено критичну проблему інформаційного перевантаження покупців через високу технічну складність та динамічність асортименту товарів.

– Розглянуто математичний апарат класичних методів фільтрації інформації, зокрема алгоритми обчислення косинусної схожості векторів у багатовимірному просторі ознак та логіку пошуку метричних сусідів.

– Висвітлено теоретичні основи побудови інтелектуальних рекомендаційних систем, їх класифікацію на статичні та персоналізовані сервіси, а також сформовано методологію кількісного оцінювання якості прогнозів за допомогою імовірнісних метрик точності й повноти.

– Проаналізовано переваги та функціональні обмеження існуючих платформ-аналогів та ізольованих алгоритмів фільтрації, що дозволило науково обґрунтувати доцільність застосування зваженої каскадної гібридної архітектури.

У другому розділі кваліфікаційної роботи:

– Досліджено ключові бізнес-процеси інформаційної системи інтернет-магазину, визначено функціональні межі взаємодії акторів та деталізовано логіку міжмодульного обміну даними у режимі реального часу.

– Обґрунтовано вибір класичної трирівневої архітектури системи та виконано процедуру нормалізації реляційної структури бази даних до третьої нормальної форми задля забезпечення цілісності та мінімізації надлишковості інформації.

– Сформовано вичерпну інженерну документацію проєкту за допомогою методів об’єктно-орієнтованого UML-моделювання й формалізовано послідовний конвеєр обробки даних підсистеми у вигляді детальної блок-схеми алгоритму.

У третьому розділі кваліфікаційної роботи:

– Розроблено повнофункціональний вебресурс електронної комерції на базі PHP-фреймворку Laravel та СУБД MySQL із гнучкою денормалізованою структурою зберігання технічних специфікацій комплектуючих у форматі JSON та модулем асинхронного клієнт-серверного логування поведінкових чинників через JavaScript Fetch API.

– Запропоновано та програмно інтегровано обчислювальні модулі колаборативного аналізу на базі алгоритму k-найближчих сусідів та предикатної контентної фільтрації брендів, об’єднані в каскадне гібридне ядро з ваговими коефіцієнтами регуляризації, що дозволило динамічно нівелювати проблему холодного старту.

– Спроєктовано та впроваджено комплекс інженерних методів оптимізації продуктивності, які знизили алгоритмічну складність обчислень «на льоту».

– Протестовано роботу рекомендаційного модуля на сформованій контрольно-тестовій вибірці зі 100 транзакцій, у результаті чого експериментально підтверджено високу ефективність гібридної моделі, яка досягла пікової точності Precision@1 = 0.80 та середнього показника 72% при мінімальному часі відгуку сервера в діапазоні 120–150 мс.

У розділі «Безпека життєдіяльності, основи охорони праці» здійснено системний аналіз умов життєдіяльності адміністратора вебресурсу в межах трикомпонентної структури «людина – машина – середовище існування» та

визначено технічні джерела потенційних небезпек. висвітлено та обґрунтовано комплексні санітарно-гігієнічні вимоги до організації й обладнання робочих місць із відеодисплейними терміналами згідно зі стандартами, що забезпечує мінімізацію статичних навантажень, захист зору та повну безпеку експлуатації створеного програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Palka O., Dmytrotsa L., Kozbur H. and Nebesnyi R. Smart people: the role of big data analytics in digital transformation [Електронний ресурс]: Режим доступу: https://ceur-ws.org/Vol-4159/paper14.pdf.

2. Palka O., Melnyk A., Dmytrotsa L., Vasylenko Y., and Klymuk N. Dynamic test case prioritisation for mobile applications based on real user behaviour data [Електронний ресурс]: Режим доступу: https://ceur-ws.org/Vol-4057/paper12.pdf.

3. Документація інструменту керування залежностями Composer для мови PHP [Електронний ресурс]: Режим доступу: https://getcomposer.org/doc/.

4. Документація системи збірки та вебфреймворку Bootstrap [Електронний ресурс]: Режим доступу: https://getbootstrap.com/docs/.

5. Документація фреймворку Laravel. Концепції архітектури та життєвий цикл запиту [Електронний ресурс]: Режим доступу: https://laravel.com/docs/.

6. Документація фреймворку Laravel. Робота з об’єктно-реляційним відображенням Eloquent ORM та зв’язками баз даних [Електронний ресурс]: Режим доступу: https://laravel.com/docs/eloquent.

7. ДСТУ ISO/IEC 12207:2015. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення [Електронний ресурс]: Режим доступу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=63451.

8. Офіційне керівництво з шаблонів проєктування та архітектурних рішень у веброзробці [Електронний ресурс]: Режим доступу: https://refactoring.guru/uk/design-patterns.

9. Відкритий репозиторій інженерних алгоритмів та структур даних для мови PHP [Електронний ресурс]: Режим доступу: https://github.com/TheAlgorithms/PHP.

10. Косинусна міра схожості та її застосування у багатовимірних метричних просторах ознак [Електронний ресурс]: Режим доступу: https://towardsdatascience.com/cosine-similarity-explained.

11. Керівництво вебдокументації MDN. Асинхронний JavaScript та робота з Fetch API [Електронний ресурс]: Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

12. Керівництво СУБД MySQL. Оптимізація запитів та побудова B-Tree індексів [Електронний ресурс]: Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/optimization.html.

13. Керівництво СУБД MySQL. Робота з типом даних JSON та вбудованими функціями обробки документів [Електронний ресурс]: Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/json.html.

14. Мартин Фаулер. Рефакторинг. Поліпшення дизайну існуючого програмного коду [Електронний ресурс]: Режим доступу: https://martinfowler.com/books/refactoring.html.

15. Ніксон Р. Створюємо динамічні вебсайти за допомогою PHP, MySQL, JavaScript, CSS та HTML5 [Електронний ресурс]: Режим доступу: https://archive.org/details/learning-php-mysql-javascript-css-html-5.

16. Офіційне керівництво з мови програмування PHP. Довідник та функції гіпертекстового препроцесора [Електронний ресурс]: Режим доступу: https://www.php.net/manual/uk/.

17. Оцінювання точності рекомендаційних сервісів за допомогою метрик зрізу Top-K та показника Precision@K [Електронний ресурс]: Режим доступу: https://towardsdatascience.com/evaluating-recommender-systems.

18. Документація інструментів проєктування та керування схемами баз даних у фреймворках [Електронний ресурс]: Режим доступу: https://laravel.com/docs/migrations.

19. Специфікація розробки та проектування архітектури модулів кошика для e-commerce систем [Електронний ресурс]: Режим доступу: https://github.com/hardevine/LaravelShoppingcart.

20. Посібник бібліотеки Scikit-learn. Алгоритми пошуку найближчих сусідів та метрики відстаней [Електронний ресурс]: Режим доступу: https://scikit-learn.org/stable/modules/neighbors.html.

21. Предикатний аналіз та колаборативна фільтрація в інтелектуальних інформаційних системах управління продажами [Електронний ресурс]: Режим доступу: https://colins.org.ua/proceedings/.

22. Річчі Ф., Рокач Л., Шапіра Б. Довідник з рекомендаційних систем [Електронний ресурс]: Режим доступу: https://link.springer.com/book/10.1007/978-1-4899-7637-6.

23. Документація інтеграції зовнішніх API та обробки транзакцій електронної комерції [Електронний ресурс]: Режим доступу: https://stripe.com/docs/api.

24. Головний репозиторій вихідного коду та архітектурного ядра фреймворку Laravel [Електронний ресурс]: Режим доступу: https://github.com/laravel/framework.

25. Документація інструментів математичного аналізу, векторизації та обробки масивів даних [Електронний ресурс]: Режим доступу: https://numpy.org/doc/.

26. Шелдон Р., Мойє Т. Навчальний посібник з мови структурованих запитів SQL та реляційних баз даних [Електронний ресурс]: Режим доступу: https://learn.microsoft.com/uk-ua/sql/relational-databases/databases/databases.

27. Репозиторій та документація спеціалізованих алгоритмів обчислення колаборативної фільтрації [Електронний ресурс]: Режим доступу: https://github.com/NicolasHug/Surprise.

28. Желібо Є.П., Зацарний В.В. Безпека життєдіяльності. Підручник. – К.: Каравела, 2009.

29. Грибан В.Г., Негодченко О.В. Охорона праці. – К.: Центр учбової літератури, 2009. 209 с..

30. Методичні вказівки для написання розділу „Безпека життєдіяльності, основи охорони праці” [Електронний ресурс]: Режим доступу: https://elartu.tntu.edu.ua/handle/lib/35902.

ДОДАТКИ

Додаток А

Лістинги файлів проекту

RecommendationController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\UserAction;
use App\Models\Product;
use Illuminate\Support\Facades\DB;

class RecommendationController extends Controller
{
    // 1. Метод для асинхронного запису дій
    public function logAction(Request $request)
    {
        $weight = ['view' => 1, 'cart' => 3, 'purchase' => 5];
        $productId = $request->product_id;
        $actionType = $request->action_type;

        // Запис дії в базу MySQL для алгоритму
        UserAction::create([
            'user_id' => auth()->id() ?? 1,
            'product_id' => $productId,
            'action_type' => $actionType,
            'weight' => $weight[$actionType] ?? 1
        ]);


        if ($actionType === 'cart') {
            $product = Product::find($productId);
            if ($product) {
                $cart = session()->get('cart', []);

                // Якщо товар вже є в кошику, збільшуємо кількість
                if (isset($cart[$productId])) {
                    $cart[$productId]['quantity']++;
                } else {
                    // Якщо немає — додаємо новий рядок
                    $cart[$productId] = [
                        "name" => $product->name,
                        "quantity" => 1,
                        "price" => $product->price
                    ];
                }
                session()->put('cart', $cart);
            }
        }
```

```
        return response()->json(['status' => 'success']);
    }


    public static function getRecommendations()
    {
        $targetUserId = auth()->id() ?? 1; // Аналізуємо поточного
авторизованого юзера або гостя

        // Крок А: Отримуємо історію взаємодій нашого користувача
        $userActions = UserAction::where('user_id',
$targetUserId)->pluck('product_id')->toArray();

        // Якщо користувач новий і нічого не натиснув — вирішуємо
проблему "холодного старту" (беремо топ-популярні)
        if (empty($userActions)) {
            return Product::inRandomOrder()->take(4)->get();
        }

        // Крок Б: Формуємо матрицю взаємодій для розрахунку
схожості товарів.
        // Шукаємо, які ще користувачі взаємодіяли з ТИМИ Ж
товарами, що й наш цільовий юзер
        $relatedUsers = UserAction::whereIn('product_id',
$userActions)
            ->where('user_id', '!=', $targetUserId)
            ->pluck('user_id')
            ->unique();

        // Якщо інших користувачів у базі поки немає (база ще не
розрослася)
        if ($relatedUsers->isEmpty()) {
            // Рекомендуємо товари з тих же категорій, якими
цікавився користувач
            $favCategories = Product::whereIn('id', $userActions)-
>pluck('category_id')->unique();
            return Product::whereIn('category_id',
$favCategories)->whereNotIn('id', $userActions)->take(4)->get();
        }

        // Крок В: Алгоритм ранжування кандидатів.
        // Витягуємо товари, які купували схожі користувачі, та
вираховуємо їхній сумарний вaaugmented-рейтинг (Score)
        $predictions = UserAction::whereIn('user_id',
$relatedUsers)
            ->whereNotIn('product_id', $userActions) // Не
рекомендуємо те, що користувач вже бачив/купив
            ->select('product_id', DB::raw('SUM(weight) as
total_weight'))
            ->groupBy('product_id')
            ->orderByDesc('total_weight')
            ->take(4)
```

```
            ->get();

        $recommendedIds = $predictions->pluck('product_id');

        // Якщо знайдено менше 4 рекомендацій, добираємо
випадковими товарами для заповнення інтерфейсу
        if ($recommendedIds->count() < 4) {
            $needed = 4 - $recommendedIds->count();
            $randomIds = Product::whereNotIn('id', $userActions)
                ->whereNotIn('id', $recommendedIds)
                ->inRandomOrder()
                ->take($needed)
                ->pluck('id');
            $recommendedIds = $recommendedIds->merge($randomIds);
        }

        // Повертаємо повноцінні об'єкти товарів з бази даних
        return Product::whereIn('id', $recommendedIds)->get();
    }
}
```

Product.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    use HasFactory;

    protected $guarded = [];


    protected $casts = [
        'specifications' => 'array',
    ];

    // Зв'язок: товар належить до однієї категорії
    public function category()
    {
        return $this->belongsTo(Category::class);
    }
}
```

UserAction.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class UserAction extends Model
{
    use HasFactory;

    public $timestamps = false;
    protected $guarded = []; // Дозволяємо запис усіх полів
}
```

CatalogController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Product;
use App\Models\Category;
use App\Http\Controllers\RecommendationController;

class CatalogController extends Controller
{
    // Головна сторінка
    public function index()
    {
        $categories = Category::all();
        $products = Product::with('category')->get();
        $recommended =
RecommendationController::getRecommendations();

        // Обовязково передаємо 'recommended' у шаблон
        return view('welcome', compact('products', 'categories',
'recommended'));
    }

    // Фільтрація товарів за категорією
    public function category($slug)
    {
        $categories = Category::all();
        $currentCategory = Category::where('slug', $slug)-
>firstOrFail();

        $products = Product::with('category')
            ->where('category_id', $currentCategory->id)
```

```
            ->get();

        $recommended =
RecommendationController::getRecommendations();

        // І сюди теж додаємо 'recommended'
        return view('welcome', compact('products', 'categories',
'currentCategory', 'recommended'));
    }

    // Детальна сторінка товару (картка)
    public function show($id)
    {
        $product = Product::with('category')->findOrFail($id);
        $currentBrand = $product->specifications['brand'] ?? null;

        $similarProducts = Product::where('category_id', $product-
>category_id)
            ->where('id', '!=', $product->id)
            ->where('specifications->brand', $currentBrand)
            ->take(3)
            ->get();

        return view('product', compact('product',
'similarProducts'));
    }
}
```

routes/web.php

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\CatalogController;
use App\Http\Controllers\RecommendationController;
use App\Http\Controllers\AuthController;
use App\Http\Controllers\CartController;
use App\Http\Controllers\AdminController;

// Каталог
Route::get('/', [CatalogController::class, 'index']);
Route::get('/category/{slug}', [CatalogController::class,
'category']);
Route::get('/product/{id}', [CatalogController::class, 'show']);

// Авторизація та Реєстрація
Route::get('/login', function () { return view('login'); })-
>name('login');
Route::post('/login', [AuthController::class, 'login']);
Route::post('/logout', [AuthController::class, 'logout']);

Route::get('/register', function () { return view('register'); });
// Сторінка реєстрації
```

```
Route::post('/register', [AuthController::class, 'register']);
 // Обробка реєстрації

// Кошик
Route::get('/cart', [CartController::class, 'index']);
Route::post('/cart/checkout', [CartController::class,
'checkout']);

// Рекомендації та Адмінка
Route::post('/api/log-action', [RecommendationController::class,
'logAction']);
Route::get('/admin', [AdminController::class, 'index']);
Route::post('/admin/product', [AdminController::class, 'store']);
```

product.blade.php

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>{{ $product->name }} - PC Store</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst
rap.min.css" rel="stylesheet">
</head>
<body class="bg-light" onload="logUserAction({{ $product->id }},
'view')"> <nav class="navbar navbar-expand-lg navbar-dark bg-dark
mb-4">
        <div class="container">
            <a class="navbar-brand" href="/">💻 PC Store</a>
            <a href="/" class="btn btn-outline-light btn-sm">← На
головну</a>
        </div>
    </nav>

    <div class="container">
        <div class="row bg-white p-4 rounded shadow-sm mb-5">
            <div class="col-md-6">
                <span class="badge bg-secondary mb-2">{{ $product-
>category->name }}</span>
                <h2>{{ $product->name }}</h2>
                <p class="text-muted fs-5">{{ $product-
>description }}</p>
                <h3 class="text-success my-4">{{
number_format($product->price, 0, ',', ' ') }} ₴</h3>

                <button onclick="logUserAction({{ $product->id }},
'cart')" class="btn btn-primary btn-lg w-50">Додати в
кошик</button>
            </div>
```

```
            <div class="col-md-6 border-start ps-4">
                <h4>Технічні характеристики (Аналіз JSON)</h4>
                <table class="table table-striped mt-3">
                    <tbody>
                        @foreach($product->specifications as $key
=> $value)
                        <tr>
                            <td class="text-muted"><strong>{{
ucfirst($key) }}</strong></td>
                            <td>{{ $value }}</td>
                        </tr>
                        @endforeach
                    </tbody>
                </table>
            </div>
        </div>

        <div class="p-4 mb-5 bg-primary bg-opacity-10 rounded
border border-primary">
            <h4 class="text-primary mb-3">🔍 Схожі моделі
(Контентна фільтрація за брендом)</h4>
            <div class="row">
                @forelse($similarProducts as $sim)
                <div class="col-md-4">
                    <div class="card h-100 shadow-sm">
                        <div class="card-body">
                            <h6><a href="/product/{{ $sim->id }}"
class="text-decoration-none text-dark">{{ $sim->name }}</a></h6>
                            <h5 class="text-primary">{{
number_format($sim->price, 0, ',', ' ') }} ₴</h5>
                        </div>
                    </div>
                </div>
                @empty
                <div class="col-12 text-muted">Схожих товарів
цього бренду наразі немає на складі.</div>
                @endforelse
            </div>
        </div>
    </div>

    <script>
        function logUserAction(productId, actionType) {
            let token = document.querySelector('meta[name="csrf-
token"]').getAttribute('content');
            fetch('/api/log-action', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json', 'X-
CSRF-TOKEN': token },
                body: JSON.stringify({ product_id: productId,
action_type: actionType })
            }).then(response => {
```

```
                if(response.ok && actionType === 'cart') {
                    alert('Товар додано в кошик! Дія покупця
записана.');
                }
            });
        }
    </script>
</body>
</html>
```

welcome.blade.php

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>PC Store - Інтернет-магазин</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst
rap.min.css" rel="stylesheet">
</head>
<body class="bg-light">

    <nav class="navbar navbar-expand-lg navbar-dark bg-dark mb-4">
        <div class="container">
            <a class="navbar-brand" href="/">💻 PC Store</a>
            <div class="d-flex text-white align-items-center">
                <a href="/cart" class="nav-link me-3">🛒 Кошик ({{
session('cart') ? array_sum(array_column(session('cart'),
'quantity')) : 0 }})</a>

                @auth
                    @if(auth()->user()->role === 'admin')
                        <a href="/admin" class="btn btn-warning
btn-sm me-3">⚙ Адмінка</a>
                    @endif

                    <span class="navbar-text text-white me-3">👤 {{
auth()->user()->name }}</span>
                    <form action="/logout" method="POST" class="d-
inline">
                        @csrf
                        <button type="submit" class="btn btn-
outline-light btn-sm">Вийти</button>
                    </form>
                @else
<a href="/login" class="btn btn-outline-light btn-sm me-
2">Увійти</a>
```

```
     <a href="/register" class="btn btn-warning btn-sm text-
dark">Реєстрація</a>
                @endauth
            </div>
        </div>
    </nav>

    <div class="container">
        @if(session('success'))
    <div class="alert alert-success my-3 shadow-sm">
        {{ session('success') }}
    </div>
@endif
        <div class="p-4 mb-5 bg-success bg-opacity-10 rounded
border border-success">
            <h4 class="text-success mb-3">□ Рекомендовано для вас
(Аналіз попиту)</h4>
            <div class="row">
                @foreach($recommended as $rec)
                <div class="col-md-3">
                    <div class="card h-100 shadow-sm border-
success">
                        <div class="card-body">
                            <span class="badge bg-success mb-
2">Хіт</span>
                            <h6 class="card-title">{{ $rec->name
}}</h6>
                            <h5 class="text-success">{{
number_format($rec->price, 0, ',', ' ') }} ₴</h5>
                        </div>
                    </div>
                </div>
                @endforeach
            </div>
        </div>

        <div class="row">

            <div class="col-md-3 mb-4">
                <div class="card shadow-sm border-0">
                    <div class="card-body">
                        <h5 class="card-title mb-3">
Категорії</h5>
                        <div class="list-group">
                            <a href="/" class="list-group-item
list-group-item-action {{ isset($currentCategory) ? '' : 'active'
}}">
                                Усі комплектуючі
                            </a>

                            @foreach($categories as $cat)
                                <a href="/category/{{ $cat->slug
}}"
```

```
                                    class="list-group-item list-
group-item-action {{ (isset($currentCategory) && $currentCategory-
>id == $cat->id) ? 'active' : '' }}">
                                        {{ $cat->name }}
                                    </a>
                                @endforeach
                            </div>
                        </div>
                    </div>
                </div>

                <div class="col-md-9">
                    <h3 class="mb-4">
                        {{ isset($currentCategory) ? 'Категорія: ' .
$currentCategory->name : 'Усі товари' }}
                    </h3>

                    <div class="row">
                        @forelse($products as $product)
                        <div class="col-md-4 mb-4">
                            <div class="card h-100 shadow-sm">
                                <div class="card-body">
                                    <span class="badge bg-secondary
mb-2">{{ $product->category->name }}</span>
                                    <h5 class="card-title">
                                        <a href="/product/{{ $product-
>id }}" class="text-decoration-none text-dark">
                                            {{ $product->name }}
                                        </a>
                                    </h5>
                                    <p class="card-text text-muted
small">{{ $product->description }}</p>
                                    <hr>
                                    <ul class="list-unstyled small mb-
3">
                                        @foreach($product-
>specifications as $key => $value)
                                            <li><strong>{{
ucfirst($key) }}:</strong> {{ $value }}</li>
                                        @endforeach
                                    </ul>
                                </div>
                                <div class="card-footer bg-white
border-top-0 text-center">
                                    <h4 class="text-dark mb-3">{{
number_format($product->price, 0, ',', ' ') }} ₴</h4>
                                    <button onclick="logUserAction({{
$product->id }}, 'cart')" class="btn btn-outline-primary w-100">До
кошика</button>
                                </div>
                            </div>
                        </div>
                        @empty
```

```
                    <div class="col-12">
                        <div class="alert alert-info">Товарів у
цій категорії тимчасово немає на складі.</div>
                    </div>
                    @endforelse
                </div>
            </div>

        </div> </div>

    <script>
        function logUserAction(productId, actionType) {
            // Отримуємо CSRF токен з мета-тегів для безпечного
POST-запиту
            let token = document.querySelector('meta[name="csrf-
token"]').getAttribute('content');

            // Асинхронна відправка події взаємодії користувача на
сервер
            fetch('/api/log-action', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                    'X-CSRF-TOKEN': token
                },
                body: JSON.stringify({
                    product_id: productId,
                    action_type: actionType
                })
            }).then(response => {
                if(response.ok) {
                    alert('Товар додано! Ваша дія записана в
матрицю для алгоритму.');
                    location.reload(); // Перезавантаження для
миттєвого перерахунку блоку рекомендацій
                }
            });
        }
    </script>
</body>
</html>
```

CartController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\UserAction;

class CartController extends Controller
```

```
{
    // Відображення сторінки кошика
    public function index()
    {
        $cart = session()->get('cart', []);
        return view('cart', compact('cart'));
    }

    // Оформлення замовлення (Симуляція покупки з перевіркою
авторизації)
    public function checkout()
    {
        // ЗАХИСТ: Якщо користувач НЕ авторизований —
перенаправляємо на сторінку входу
        if (!auth()->check()) {
            return redirect('/login')->withErrors([
                'email' => 'Для оформлення замовлення та фіксації
попиту необхідно увійти у свій обліковий запис.',
            ]);
        }

        $cart = session()->get('cart', []);

        if (empty($cart)) {
            return redirect('/')->with('error', 'Кошик
порожній!');
        }

        // Для кожного товару в кошику записуємо подію PURCHASE
(вага 5)
        foreach ($cart as $productId => $details) {
            UserAction::create([
                'user_id' => auth()->id() ?? 1,
                'product_id' => $productId,
                'action_type' => 'purchase',
                'weight' => 5
            ]);
        }

        // Очищаємо кошик після покупки
        session()->forget('cart');

        // Повертаємося на головну з повідомленням успіху
        return redirect('/')->with('success', 'Замовлення успішно
оформлено! Підсистема рекомендацій врахувала вашу покупку.');
    }
}
```

Файл конфігурації середовища .env

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:5fnbkUI9WadnGGyan/JwQhejs6VweqYa9qy4Taiq51A=
APP_DEBUG=true
```

```
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=computer_store_db
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```