МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет економіки, підприємництва та інформаційних технологій
Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2026 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ З ІНТЕЛЕКТУАЛЬНИМ МОДУЛЕМ РЕКОМЕНДАЦІЙ

Виконав – студент 4 курсу, групи 41-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
Кузьмін Вадим Олександрович

Науковий керівник – к.т.н., доцент
Ліщук Роман Ігорович

Умань – 2026

Зміст

ВСТУП 3

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ ОБ'ЄКТА ДОСЛІДЖЕННЯ 5

1.1. Аналіз предметної області 5

1.2. Порівняльний аналіз об'єкта досліджень 6

1.3. Постановка задачі 11

РОЗДІЛ 2 ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ З ІНТЕЛЕКТУАЛЬНИМ МОДУЛЕМ РЕКОМЕНДАЦІЙ 14

2.1. Обґрунтування стеку технологій 14

2.2. Архітектура веб-додатку для інтернет-магазину з інтелектуальним модулем рекомендацій 16

2.3. Проєктування програмного забезпечення 20

2.4. Проектування інтелектуального модуля рекомендацій 22

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ З ІНТЕЛЕКТУАЛЬНИМ МОДУЛЕМ РЕКОМЕНДАЦІЙ 26

3.1. Ралізація алгоритмів обробки інформаційних потоків та архітектури бази даних 26

3.2. Реалізація користувацького інтерфейсу 31

3.3. Тестування роботи веб-додатку та модулів програми 36

3.4. Програмна реалізація інтелектуального модуля рекомендацій .. 49

ВИСНОВКИ 54

СПИСОК ЛІТЕРАТУРИ 56

ДОДАТКИ 59

ВСТУП

Сучасна сфера електронної комерції та роздрібної торгівлі є одним із ключових секторів глобальної економіки. Високий рівень насиченості ринку різноманітними товарами зумовлює жорстку конкуренцію серед підприємств. В умовах паритетності цінових та якісних характеристик товарів вирішальним фактором для споживача стає рівень клієнтського сервісу та зручність взаємодії з торговим майданчиком. Розробка та впровадження спеціалізованих веб-додатків для електронної комерції є ефективним інструментом оптимізації процесу придбання товарів, що забезпечує дистанційне обслуговування клієнтів з високим рівнем комфорту.

Цифровізація торговельних процесів та наявність власного програмного рішення для онлайн-продажів є критично важливими факторами забезпечення конкурентоспроможності сучасних підприємств. Впровадження електронних інформаційних систем дозволяє автоматизувати рутинні операції, оптимізувати операційні витрати на персонал та збільшити загальний товарообіг компанії. З позиції споживача використання таких платформ мінімізує часові та логістичні витрати, пов'язані з фізичним пошуком і придбанням товарів, а також усуває ризики відсутності продукції на складах традиційних магазинів. Отже, розробка інформаційної системи для забезпечення електронної комерції є актуальним науково-практичним завданням, що сприяє раціональному розподілу ресурсів усіх учасників торговельного процесу.

Метою даної кваліфікаційної роботи є розробка програмного забезпечення у вигляді веб-додатку для автоматизації процесів реалізації подарункової продукції, управління асортиментом та обробки замовлень, а також інтеграція в розроблену систему інтелектуального модуля рекомендацій.

Об'єктом розробки є процес автоматизації електронної комерції, управління товарним асортиментом та обробки замовлень за допомогою веб-технологій, а також інтелектуальний модуль рекомендацій товарів.

Предметом розробки є методи та засоби створення веб-додатку на базі платформи .NET, застосування алгоритмів матричної факторизації, а також методів оцінки частоти термінів та оберненої частоти документів у контексті функціонування інтелектуального модуля рекомендацій.

Для досягнення поставленої мети сформульовано наступні завдання:

- провести системний аналіз предметної області;
- здійснити порівняльний аналіз існуючих програмних аналогів;
- сформувати специфікацію функціональних вимог до інформаційної системи;
- виконати архітектурне та логічне проектування програмного рішення;
- програмно реалізувати розроблюваний додаток;
- провести комплексне тестування створеного програмного забезпечення.

Отримані результати формують комплексну методологічну та практичну базу для проектування і впровадження спеціалізованих інформаційних систем. Розроблений програмний продукт є готовим до експлуатації рішенням, що може бути безпосередньо інтегроване у бізнес-процеси торговельних підприємств з метою підвищення ефективності їхньої діяльності та розширення каналів збуту.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ ОБ’ЄКТА ДОСЛІДЖЕННЯ

1.1. Аналіз предметної області

Сучасне суспільство характеризується високим рівнем споживання матеріальних благ, що охоплює різноманітні сфери життєдіяльності індивіда від використання високотехнологічних пристроїв до задоволення побутових потреб. Забезпечення цих потреб вимагає значних ресурсних, зокрема часових, витрат. Відповідно до статистичних даних, середньостатистичний споживач витрачає на здійснення покупок у традиційних магазинах близько 25 тисяч годин [1]. У зв'язку з цим виникає об'єктивна необхідність пошуку ефективних методів оптимізації купівельного процесу з метою мінімізації непродуктивних витрат часу.

З боку комерційного сектору пріоритетним завданням є максимізація економічної ефективності підприємницької діяльності. Досягнення високих показників рентабельності безпосередньо залежить від здатності суб'єктів господарювання адаптуватися до сучасних ринкових умов шляхом автоматизації торговельних процесів. Впровадження інформаційних систем управління продажами дозволяє суттєво пришвидшити товарообіг, оптимізувати внутрішню логістику та пропорційно збільшити кінцевий прибуток підприємства.

Універсальним рішенням, що одночасно задовольняє інтереси як споживачів, так і підприємців, є використання платформ електронної комерції. Попри значне поширення інтернет-магазинів у сучасній глобальній економіці, їхня наукова та практична актуальність продовжує неухильно зростати. Ця тенденція пояснюється високою ефективністю віртуальних

торгових майданчиків та постійним розширенням їхнього технологічного функціоналу.

Для користувачів ключовою перевагою експлуатації спеціалізованих веб-додатків є безпрецедентний рівень ергономічності та комфорту. Процес пошуку, вибору та оформлення замовлення здійснюється дистанційно через інтуїтивно зрозумілі інтерфейси, що повністю виключає необхідність витрачати час на логістику та фізичне відвідування торгових точок. Вивільнений часовий ресурс споживач має змогу раціонально перерозподілити на соціально значущі сфери життя, включаючи рекреацію, міжособистісну комунікацію та професійний саморозвиток.

Для суб'єктів підприємницької діяльності реалізація товарів через веб-додатки супроводжується кардинальним зниженням операційних витрат. Ведення комерційної діяльності у цифровому середовищі усуває необхідність фінансування дороговартісних торгових площ та дозволяє оптимізувати штат персоналу за рахунок програмної автоматизації рутинних завдань. Зазначені економічні переваги суттєво знижують бар'єри входу на ринок для новостворених компаній, а також формують сприятливе середовище для структурної модернізації та масштабування тих підприємств, які потребують оновлення власних бізнес-моделей.

1.2. Порівняльний аналіз об’єкта досліджень

Обмін подарунковою продукцією виступає важливою складовою соціальної взаємодії та комунікації індивідів у суспільстві. Традиційні святкові заходи систематично супроводжуються процесом дарування, що забезпечує належну емоційну залученість учасників. З огляду на стабільний попит, на сучасному сегменті ринку електронної комерції функціонує значна

кількість спеціалізованих інформаційних систем, орієнтованих на реалізацію відповідної категорії товарів.

Прикладом такого програмного рішення є сайт PRESENTVILLE [2] (рис. 1.1). Зазначена платформа пропонує репрезентативний асортимент товарних позицій, що охоплює предмети одягу, аксесуари та різноманітні елементи інтер'єрного декору.

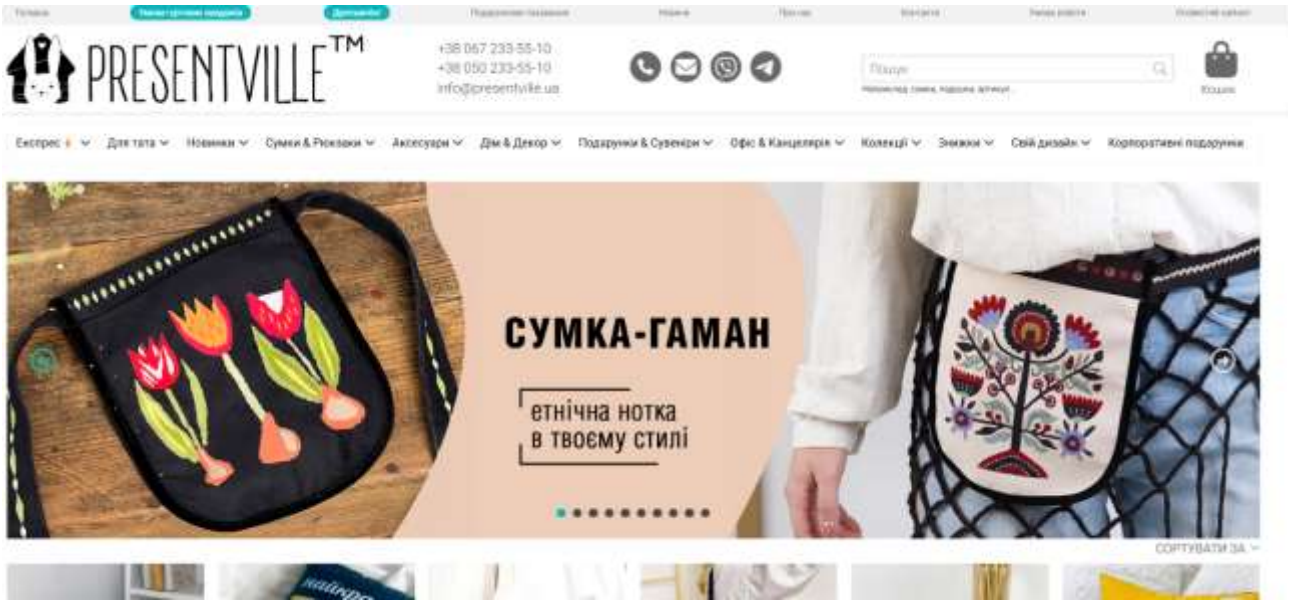


Рисунок 1.1 – Головна сторінка сайту PRESENTVILLE

До базових функціональних можливостей досліджуваного веб-додатку належать:

- ініціалізація та модифікація персонального облікового запису користувача;
- перегляд електронного каталогу продукції;
- фільтрація асортименту за текстовими запитами та ціновими критеріями;
- алгоритмічне сортування товарів за визначеними параметрами;
- формування індивідуального переліку пріоритетних товарних позицій;

- управління віртуальним кошиком покупця шляхом додавання та вилучення обраних одиниць;
- процедура оформлення електронного замовлення.

За результатами аналізу даного програмного забезпечення було виявлено певні архітектурні та функціональні обмеження, серед яких основним є відсутність підсистеми зворотного зв'язку, що повністю унеможливлює публікацію текстових рецензій та оцінювання клієнтами безпосередньо на сторінках товарів.

Наступним об'єктом порівняльного аналізу виступає платформа електронної комерції Gifty [3]. Функціональне забезпечення даної системи є практично ідентичним до попередньо розглянутого програмного рішення (рис. 1.2).

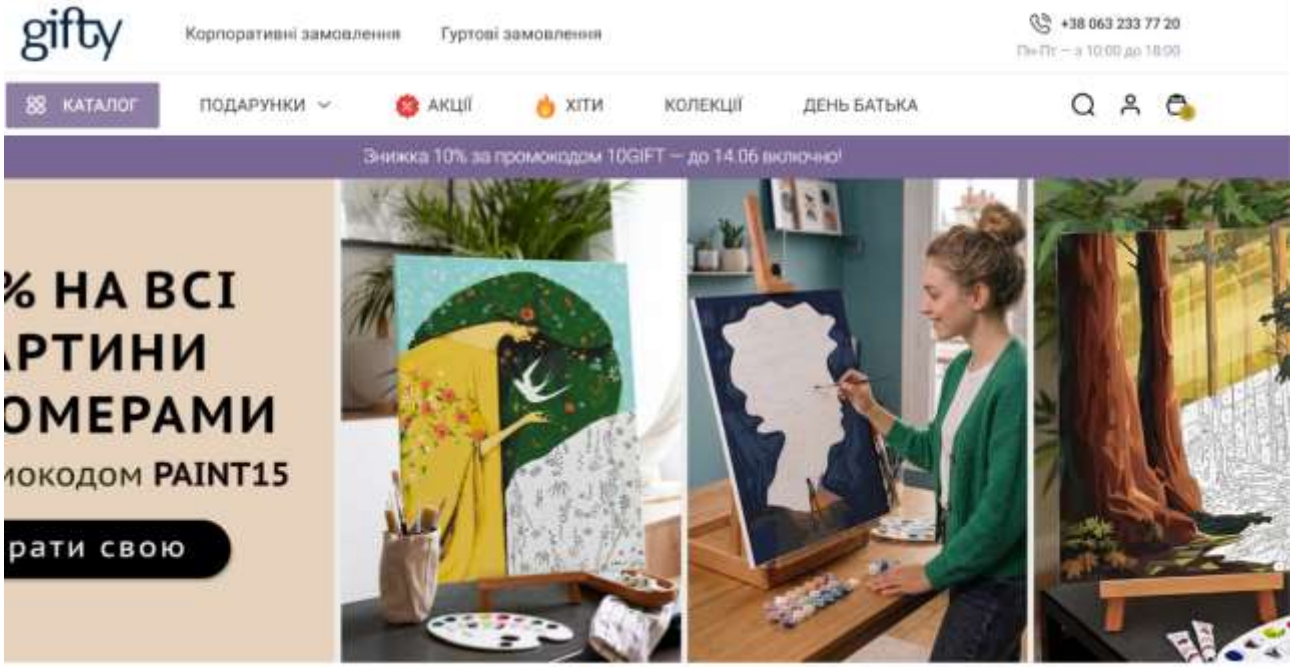


Рисунок 1.2 – Головна сторінка сайту Gifty

Відмінною характеристикою досліджуваного ресурсу є наявність інтегрованого інформаційного розділу у форматі блогу, що використовується для публікації фахових матеріалів та новинних зведень. Проте, аналогічно до першої розглянутої платформи, суттєвим недоліком залишається відсутність підсистеми зворотного зв'язку для розміщення користувацьких рецензій на продукцію.

Окремим предметом дослідження виступає спеціалізований веб-додаток Kashalot Gift [4] . Специфіка даної системи полягає у реалізації виключно концептуальної подарункової продукції (рис. 1.3). Архітектура електронного каталогу відзначається структурною заміною стандартних товарних категорій на тематичні напрямки, де базовими одиницями реалізації є комплексні набори відповідного цільового наповнення. Серед додаткових експлуатаційних переваг необхідно виокремити наявність вбудованого інструментарію для оперативного комунікаційного зв'язку зі службою клієнтської підтримки у режимі реального часу. Базова алгоритмічна архітектура та набір стандартних можливостей системи є повністю еквівалентними до раніше проаналізованих програмних аналогів.



Рисунок 1.3 – Головна сторінка сайту Kashalot Gift

Проведений системний аналіз існуючих програмних рішень у сегменті електронної комерції подарунковою продукцією дозволив сформувати комплексне уявлення про поточний стан предметної області. Більшість розглянутих інформаційних систем володіють достатнім базовим функціоналом, який забезпечує виконання стандартних торговельних операцій, таких як навігація за електронним каталогом, фільтрація асортименту, управління віртуальним кошиком та дистанційне оформлення замовлень. Деякі платформи впроваджують специфічні структурні та комунікаційні рішення, зокрема тематичне групування товарних позицій, ведення інформаційних блогів та інтеграцію засобів оперативного зв'язку зі службою клієнтської підтримки.

Разом з тим, порівняльне дослідження виявило ряд суттєвих недоліків, притаманних наявним на ринку веб-додаткам. Основним функціональним обмеженням є відсутність повноцінної підсистеми зворотного зв'язку, що повністю унеможливлює публікацію клієнтських відгуків, об'єктивне оцінювання продукції та формування соціального рейтингу товарів. Крім того, проаналізовані аналоги позбавлені сучасних алгоритмічних інструментів персоналізації користувацького досвіду.

З огляду на виявлені функціональні прогалини, обґрунтовується науково-практична доцільність створення нового конкурентоспроможного програмного забезпечення. Розроблювана інформаційна система повинна усунути виявлені обмеження шляхом впровадження підсистеми рецензування та оснащення платформи інтелектуальним модулем рекомендацій. Реалізація такого підходу дозволить не лише автоматизувати процес підбору релевантних товарів для кожного окремого користувача, але

й суттєво підвищити загальний рівень клієнтського сервісу та комерційну ефективність розроблюваного веб-додатку.

1.3. Постановка задачі

Головним завданням розробки є проектування та впровадження спеціалізованого веб-ресурсу, призначеного для оптимізації процесу дистанційного придбання подарункової продукції споживачами, а також для забезпечення автоматизованого обліку товарного асортименту та ефективного управління обробкою замовлень з боку адміністративного персоналу. Архітектура безпеки та управління доступом розроблюваної інформаційної системи передбачає реалізацію підсистеми автентифікації та авторизації користувачів на основі унікальних облікових даних. Програмне рішення ґрунтується на рольовій моделі доступу, яка диференціює повноваження суб'єктів системи на рівні неавторизованого відвідувача, зареєстрованого користувача, менеджера та адміністратора.

Функціональна специфікація для неавторизованих суб'єктів доступу включає базовий інструментарій для взаємодії з електронним каталогом. Зокрема, передбачено можливості перегляду товарних позицій та клієнтських рецензій, застосування алгоритмів сортування за ціновими та категорійними критеріями, а також здійснення контекстного пошуку за найменуванням продукції. Крім того, неавторизовані відвідувачі наділяються правами управління віртуальним кошиком, що охоплює додавання, вилучення та кількісне коригування обраних одиниць, з подальшою можливістю ініціалізації процедури оформлення замовлення та створення персонального облікового запису.

Обліковий запис зареєстрованого користувача розширює базовий функціонал можливостями персоналізації та активної взаємодії з

платформою. Після проходження процедури автентифікації суб'єкт отримує доступ до управління власним профілем, що включає редагування персональних даних, оновлення графічного ідентифікатора та модифікацію параметрів безпеки. Важливою складовою є надання прав на участь у формуванні соціального рейтингу платформи шляхом публікації та видалення власних відгуків, позитивного чи негативного оцінювання товарів, а також взаємодії з інтелектуальним модулем рекомендацій для отримання персоналізованих пропозицій.

Роль менеджера системи акумулює користувацькі повноваження та доповнюється специфічними інструментами операційного управління. До компетенції даної категорії персоналу належить модерація клієнтського контенту шляхом вилучення невідповідних коментарів, а також забезпечення повного циклу обробки запитів через моніторинг загального реєстру замовлень та зміну їхніх статусних станів. Найвищий рівень привілеїв у системі закріплено за роллю адміністратора, який додатково здійснює комплексне адміністрування платформи. Адміністративний інструментарій забезпечує можливість глибокої модифікації електронного каталогу через управління товарними позиціями та їхньою категорійною структурою, а також здійснює глобальний контроль над клієнтською базою, що передбачає моніторинг, пошук, сортування профілів та перепризначення рольових політик доступу.

Архітектура користувацького інтерфейсу веб-додатку проектується з урахуванням ієрархії ролей та відповідного розмежування доступу до візуальних компонентів. Базова структура маршрутизації для відвідувачів охоплює екрани реєстрації та автентифікації, загальний каталог, деталізовані картки товарів, інтерфейси віртуального кошика та процедури оформлення покупки, а також стандартну сторінку обробки помилок навігації. Для

авторизованих суб'єктів цей перелік розширюється сторінкою персонального профілю. Операційна діяльність менеджерів забезпечується спеціалізованим інтерфейсом інформаційної дошки замовлень, тоді як для адміністраторів розробляється комплексна панель управління всією інформаційною системою.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ З ІНТЕЛЕКТУАЛЬНИМ МОДУЛЕМ РЕКОМЕНДАЦІЙ

2.1. Обґрунтування стеку технологій

Для програмної реалізації інформаційної системи було обрано комплексний стек технологій, що забезпечує високу продуктивність, масштабованість та надійність розроблюваного веб-додатку. В якості сховища даних використовується Microsoft SQL Server [5] – реляційна система управління базами даних, призначена для структурованого зберігання інформаційного масиву системи. Взаємодія з даними здійснюється за допомогою декларативної мови запитів Transact-SQL. До ключових переваг даної СКБД належать високий рівень масштабованості, оптимізовані механізми розгортання та інтеграції великих обсягів даних, висока продуктивність обчислень, надійність та безпека збереження інформації, а також ергономічність експлуатації.

Програмним інструментом для доступу до бази даних виступає Entity Framework Core [6], що реалізує технологію об'єктно-реляційного відображення. Це забезпечує рівень абстракції над сховищем даних та дозволяє маніпулювати сутностями безпосередньо як об'єктами класів. Архітектура фреймворку підтримує інтеграцію з різноманітними СКБД за наявності відповідних провайдерів. Експлуатаційні переваги інструменту включають мінімізацію витрат часу та ресурсів на розробку, можливість візуального проектування моделей даних, автоматичну генерацію програмного коду, уніфікацію синтаксису запитів незалежно від типу джерела інформації, а також кросплатформність.

Серверна частина системи базується на ASP.NET Core [7] – програмному фреймворку для побудови веб-додатків. У межах даного

дослідження технологія застосовується для розробки прикладного програмного інтерфейсу на основі архітектурного стилю REST, де ресурси репрезентуються через методи контролерів, а взаємодія відбувається за допомогою базових методів протоколу HTTP. Серед основних переваг виокремлюються кросплатформність, висока швидкодія, гнучкість у виборі середовища розгортання та вбудована підтримка інверсії управління через впровадження залежностей. Крім того, на серверній стороні інтегровано ML.NET [8] – відкриту бібліотеку машинного навчання, яка застосовується для побудови та навчання предиктивних моделей, що функціонують спільно з інтелектуальним модулем рекомендацій. Її використання обґрунтоване кросплатформністю та нативною інтеграцією з базовою платформою розробки, що нівелює необхідність залучення додаткових інструментів та оптимізує процес програмної реалізації.

Клієнтська частина веб-додатку реалізована за допомогою ReactJS [9] – відкритої бібліотеки мови програмування JavaScript, призначеної для побудови користувацьких інтерфейсів. До її експлуатаційних переваг належать висока швидкодія, архітектурна гнучкість завдяки модульному підходу, оптимізована продуктивність за рахунок використання механізму віртуальної об'єктної моделі документа та загальна ергономічність. Для прогнозованого управління станом програмного забезпечення на стороні клієнта застосовується ReduxJS [10]. Впровадження цієї технології для організації глобального інформаційного сховища та централізованого маніпулювання станом додатку зумовлене такими перевагами, як детермінованість стану системи, висока загальна продуктивність та зручність інтеграції у процесі розробки.

Комплексний аналіз та обґрунтування вибору даного стеку технологій базується на принципах архітектурної цілісності та розподілу обов'язків.

Використання єдиної екосистеми .NET для розробки серверної частини формує високооптимізоване та надійне середовище. Це дозволяє забезпечити безперебійну та швидку взаємодію між ядром бізнес-логіки, базою даних та інтелектуальним модулем рекомендацій без необхідності налаштування складних міжплатформних інтеграцій. Виділення клієнтської частини у самостійний односторінковий додаток на базі ReactJS та ReduxJS гарантує створення сучасного, реактивного та зручного користувацького інтерфейсу, що є критично важливим для сфери електронної комерції. Такий архітектурний поділ на незалежні клієнтську та серверну підсистеми сприяє високій масштабованості системи, спрощує подальший супровід програмного коду, полегшує процес тестування та дозволяє незалежно модернізувати окремі компоненти веб-додатку без порушення працездатності інформаційної системи в цілому.

2.2. Архітектура веб-додатку для інтернет-магазину з інтелектуальним модулем рекомендацій

Проектування розроблюваного веб-додатку базується на принципах клієнт-серверної архітектури [11]. Зазначена модель являє собою розподілену обчислювальну структуру, в межах якої серверний вузол здійснює централізоване розміщення, надання та адміністрування апаратних і програмних ресурсів, а також сервісів, що споживаються клієнтськими застосунками. Візуалізація базової концепції даної взаємодії наведена на рисунку 2.1.

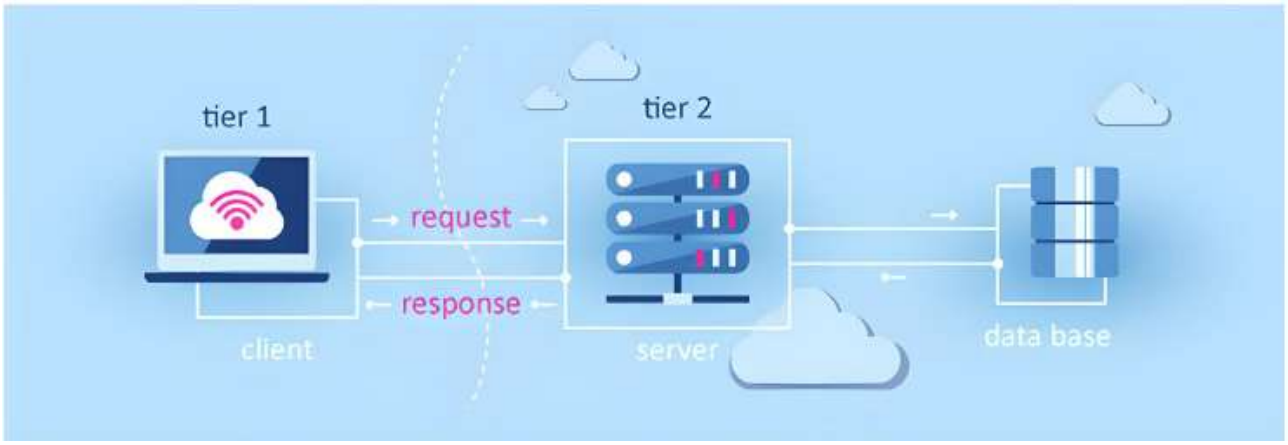


Рисунок 2.1 – Схема клієнт-серверної архітектури

Алгоритм функціонування даної моделі передбачає ініціалізацію комунікаційного сеансу з боку клієнта шляхом формування та відправлення мережевого запиту. Серверна частина здійснює прийом, маршрутизацію та обробку отриманого запиту, що за необхідності супроводжується транзакційними зверненнями до системи управління базами даних. На завершальному етапі згенерований масив даних або статус виконання операції формується у відповідь та відправляється назад до клієнтського вузла.

Раціональне проектування серверної частини виступає одним із найбільш критичних аспектів життєвого циклу розробки програмного забезпечення. Фундаментальні архітектурні рішення безпосередньо детермінують ключові експлуатаційні метрики системи, зокрема загальну продуктивність, швидкість імплементації нового функціоналу та рівень модульності кодової бази. Можливість безперешкодної заміни окремих програмних компонентів та адаптивність до майбутнього масштабування прямо пропорційно впливають на мінімізацію економічних і часових витрат. Відповідно, вибір оптимального архітектурного патерну є визначальним

етапом інженерного проектування, що дозволяє уникнути накопичення технічного боргу.

В якості методологічної основи для реалізації серверної частини було обрано концепцію чистої архітектури [12]. Фундаментальна парадигма даного підходу полягає у формуванні доменно-орієнтованої структури, де ядром програмної системи виступає виключно бізнес-логіка. На противагу традиційним моделям, де правила предметної області залежать від механізмів доступу до даних чи специфіки інфраструктурних компонентів, у чистій архітектурі застосовується принцип інверсії залежностей. Згідно з цим принципом, деталі інфраструктурної реалізації підпорядковуються базовому функціоналу додатку. Технічно це досягається шляхом декларування абстракцій та інтерфейсів у центральному ядрі системи з їхньою подальшою імплементацією на зовнішніх інфраструктурних рівнях. Графічне відображення структурних компонентів даної моделі представлено на рисунку 2.2.

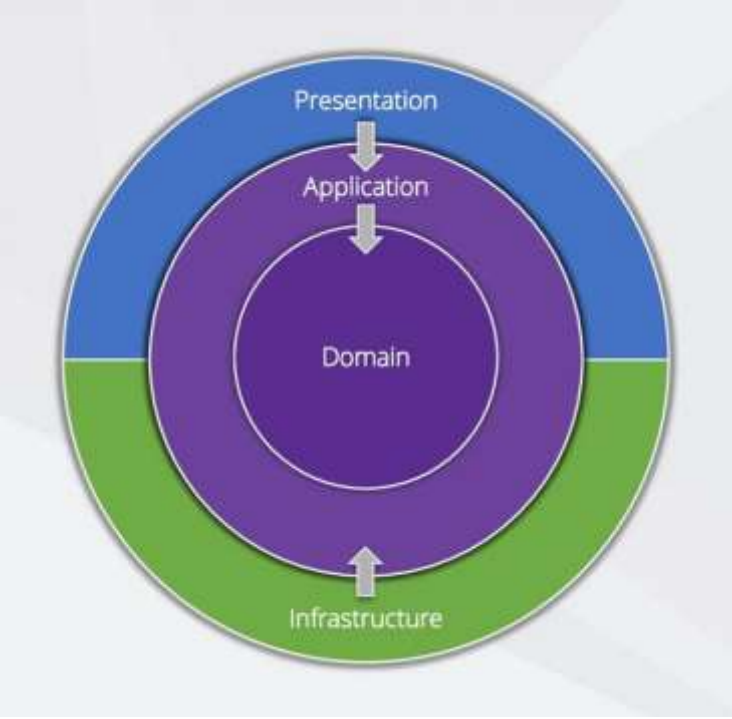


Рисунок 2.2 – Демонстрація чистої архітектури

Впровадження подібної ієрархічної структури забезпечує високий рівень абстрагування від низькорівневих деталей інфраструктури, дозволяючи акумулювати інженерні ресурси на реалізації та оптимізації безпосередньо бізнес-правил. Ефективність даного підходу стає особливо очевидною при його порівняльному аналізі з класичною багаторівневою архітектурою, яка характеризується дано-центричною моделлю з прямою залежністю бізнес-логіки від рівня доступу до даних. У межах багаторівневої парадигми будь-яка модифікація механізмів збереження інформації неминуче провокує необхідність каскадного рефакторингу коду на рівні бізнес-логіки. Використання чистої архітектури повністю нівелює ризики подібної зв'язності, що гарантує системі виняткову гнучкість та стійкість до структурних змін.

Підсумовуючи вищезазначене, доцільно виокремити наступні фундаментальні переваги застосування концепції чистої архітектури при розробці інформаційної системи:

- абсолютна незалежність ядра системи від специфіки обраної системи управління базами даних та зовнішніх програмних фреймворків;
- повна ізоляція бізнес-логіки від рівня представлення, що забезпечує можливість радикальної модифікації або повної заміни користувацького інтерфейсу без будь-якого втручання у внутрішні алгоритми функціонування системи;
- високий рівень написання коду, що значно спрощує процеси ізольованого модульного тестування компонентів предметної області без необхідності розгортання супутніх інфраструктурних елементів.

2.3. Проєктування програмного забезпечення

Процес проектування програмного забезпечення вимагає формалізованого опису функціональних вимог та специфікації поведінки майбутньої системи. Ефективним інструментом візуального моделювання системних процесів виступає діаграма прецедентів або варіантів використання [13]. Дана нотація дозволяє репрезентувати архітектуру інформаційної системи через сукупність сутностей та прецедентів, що відображають сценарії їхньої взаємодії. Прецедент визначає специфічний набір функцій, які інформаційна система надає зовнішньому суб'єкту, абстрагуючись від низькорівневих технічних деталей алгоритмічної реалізації цих процесів.

У побудованій моделі застосовуються базові типи структурних відношень: включення та розширення. Відношення включення характеризує строгу структурну залежність, за якої базовий прецедент обов'язково ініціює виконання допоміжного варіанту використання для успішного завершення свого сценарію. Відношення розширення визначає опціональну поведінку, за якої базовий прецедент може бути доповнений специфічним функціоналом за настання певних умов, формуючи альтернативний вектор розвитку системних подій.

Розроблена діаграма прецедентів наочно ілюструє поведінкову модель проектованої системи електронної комерції. Архітектура функціональної взаємодії передбачає наявність ключових акторів з диференційованими рівнями доступу: незареєстрованого користувача, зареєстрованого користувача, менеджера та адміністратора.

Незареєстрований користувач наділений базовими правами взаємодії з електронним каталогом. Цей суб'єкт може здійснювати перегляд товарних позицій, що опціонально розширюється допоміжними функціями сортування

за категоріями, найменуванням або ціновою політикою. Також передбачено можливості детального ознайомлення з конкретним товаром, перегляду клієнтських рецензій та додавання обраних номенклатурних одиниць до віртуального кошика.

Зареєстрований користувач володіє розширеним спектром можливостей, реалізація переважної більшості яких вимагає обов'язкового виконання прецеденту авторизації. До специфічних повноважень цього суб'єкта належать управління персональним профілем, що включає редагування особистої інформації, зміну пароля та оновлення аватара. Авторизований суб'єкт бере активну участь у формуванні соціального рейтингу платформи через публікацію відгуків, виставлення оцінок, а також позитивне чи негативне маркування продукції. Важливою складовою є доступ до персоналізованих пропозицій, згенерованих інтелектуальним модулем рекомендацій. Окрім цього, даний актор здійснює повноцінне управління вмістом кошика та ініціює процедуру оформлення замовлення, що передбачає обов'язкове заповнення відповідних інформаційних форм.

Роль менеджера орієнтована на операційне обслуговування торговельного майданчика. Після проходження процедури авторизації цей суб'єкт здійснює моніторинг реєстру замовлень з можливістю подальшої зміни їхніх статусних станів. Додатково менеджер виконує функції модерації контенту шляхом вилучення невідповідних користувацьких відгуків.

Адміністратор системи володіє найвищим рівнем системних привілеїв. До його виключної компетенції належить комплексний менеджмент товарного асортименту та категорійної структури, що охоплює процеси додавання, редагування та вилучення відповідних записів. Крім того, адміністратор здійснює глобальне управління клієнтською базою, що

передбачає перегляд списку користувачів та модифікацію їхніх рольових політик доступу.

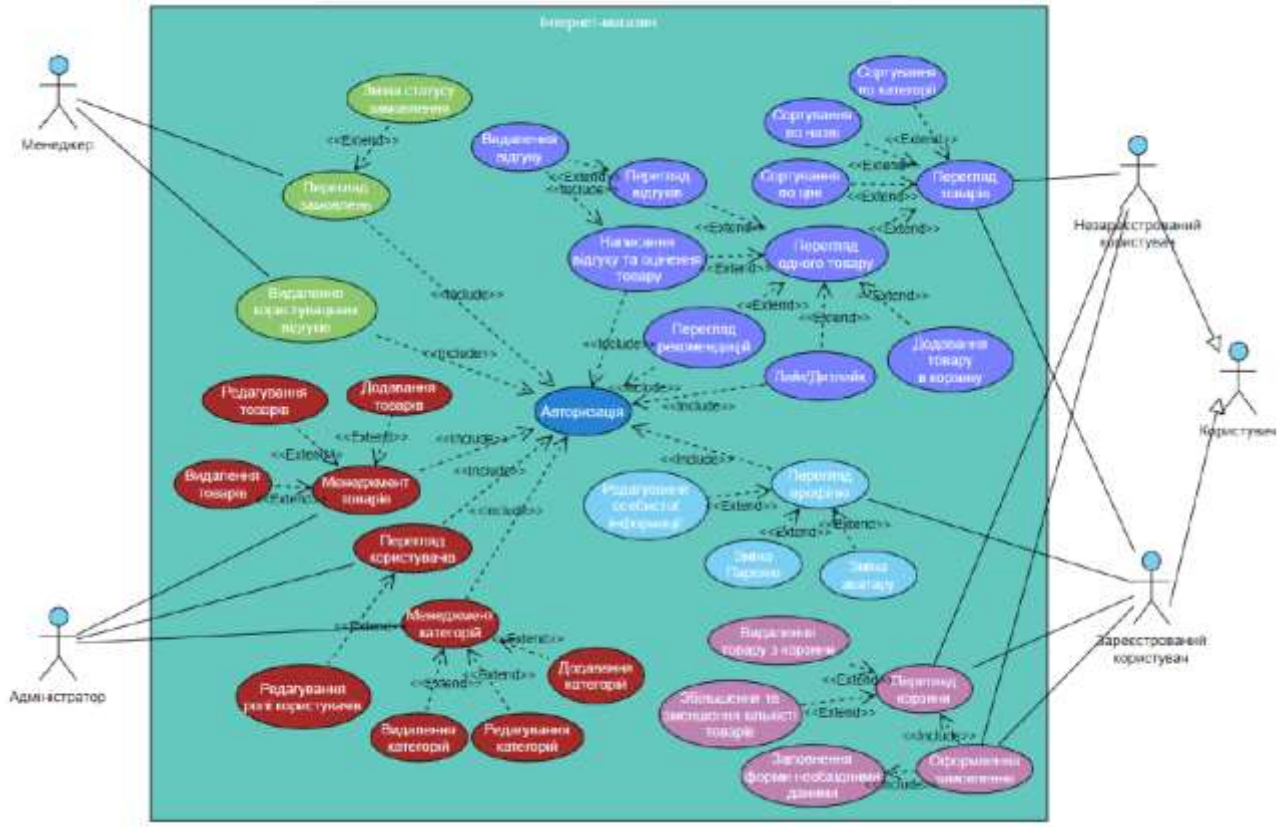


Рисунок 2.3 – Діаграма прецедентів інформаційної системи

2.4. Проектування інтелектуального модуля рекомендацій

Архітектура розроблюваного інтелектуального модуля рекомендацій базується на синергетичному поєднанні двох фундаментальних парадигм машинного навчання: колаборативної фільтрації з використанням алгоритмів матричної факторизації та контентно-орієнтованої фільтрації.

Базова гіпотеза колаборативного підходу полягає у припущенні, що суб'єкти, які демонстрували високу кореляцію преференцій у минулому, зберігатимуть подібність інтересів і в майбутніх періодах. Для алгоритмічної реалізації цього принципу формується розріджена матриця взаємодій, де

рядки відповідають користувачам, а стовпці – товарам. Елементами матриці виступають кількісні оцінки. Оскільки матриця є суттєво неповною, головним математичним завданням є апроксимація пропущених значень. З точки зору математичного обґрунтування застосовується метод матричної факторизації, який розкладає вихідну матрицю оцінок R розмірності $m \times n$ на добуток двох низькорангових матриць прихованих ознак: матриці профілів користувачів P та матриці властивостей товарів Q. Прогнозована оцінка $\hat{r}_{ui}$ користувача u для товару i обчислюється як скалярний добуток відповідних векторів:

$$\hat{r}_{ui} = p_u \cdot q_i^T. \tag{2.1}$$

Оптимізація параметрів моделі здійснюється шляхом мінімізації функції втрат, що включає середньоквадратичну похибку та члени регуляризації для запобігання перенавчанню. Товари з найвищими прогнозованими оцінками формують цільову вибірку для рекомендації.

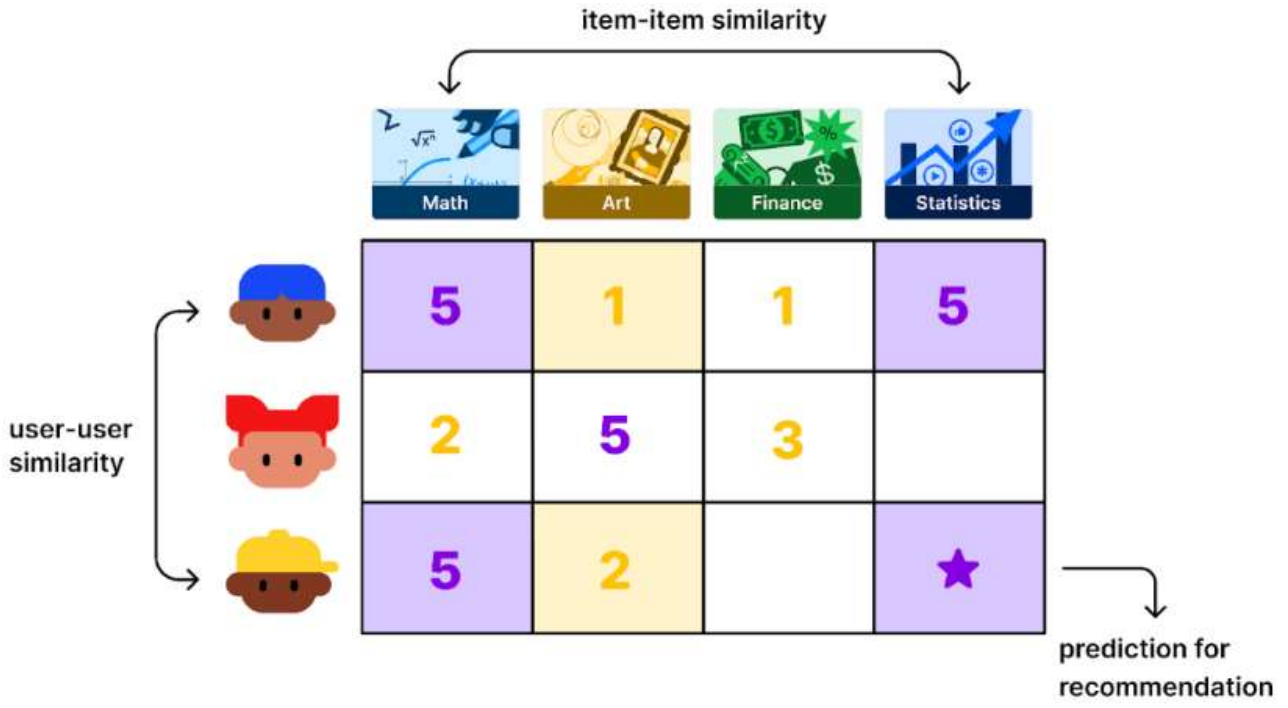


Рисунок 2.4 – Принцип роботи колаборативної фільтрації

Наведений рисунок візуалізує фундаментальний механізм колаборативної фільтрації, що полягає у виявленні неявних кластерів користувачів зі схожими патернами поведінки та прогнозуванні відсутніх зв'язків у розрідженій матриці на основі перехресного аналізу наявних рейтингів.

Ефективність описаного алгоритму прямо пропорційна щільності заповнення матриці взаємодій. Критичним обмеженням даного методу є проблема холодного старту, що виникає на початкових етапах експлуатації інформаційної системи за умови дефіциту накопичених статистичних даних, що фактично унеможливлює формування релевантних прогнозів через відсутність достатньої кількості векторів для обчислення подібності.

Для компенсації зазначеного недоліку впроваджено підсистему контентно-орієнтованої фільтрації, математичний апарат якої адаптує класичні метрики інформаційного пошуку: частоту терміна та обернену частоту документа. У контексті розроблюваної платформи метрика частоти терміна відображає інтенсивність позитивних взаємодій користувача з товарами певної категорії. Однак абсолютна частота не є достатнім критерієм релевантності, оскільки вимагає нормування на глобальний розподіл категорій у системі. Для цього застосовується метрика оберненої частоти документа, яка логарифмічно масштабує вагу категорії обернено пропорційно до її загальної поширеності у товарному каталозі. Математично вага певної категорії c для користувача u визначається як добуток цих двох компонентів:

$$w_{uc} = TF_{uc} \times IDF_c. \tag{2.2}$$

На основі обчислених вагових коефіцієнтів формується багатовимірний вектор профілю користувача, що дозволяє кількісно оцінити ймовірнісну релевантність будь-якої нової товарної позиції шляхом обчислення міри косинусної близькості між вектором інтересів клієнта та вектором категорійних характеристик продукту. Комплексне застосування цих математичних моделей дозволяє створити стійкий до відсутності даних та високоточний інтелектуальний модуль рекомендацій.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ З ІНТЕЛЕКТУАЛЬНИМ МОДУЛЕМ РЕКОМЕНДАЦІЙ

3.1. Ралізація алгоритмів обробки інформаційних потоків та архітектури бази даних

Взаємодія з масивами даних в інформаційній системі реалізується через графічний інтерфейс користувача, який забезпечує візуалізацію інформації та ініціалізацію обчислювальних процесів. Конфігурація відображення графічного інтерфейсу повністю детермінується поточним станом програмного забезпечення. Стан додатка є комплексним набором параметрів та властивостей, що акумулюють динамічні дані, необхідні для забезпечення сеансу взаємодії, зокрема переліки номенклатурних позицій або параметри поточної маршрутизації. Для централізованого управління станом клієнтської частини інтегровано технологію ReduxJS. Застосування цього інструментарію дозволяє консолідувати всю оперативну інформацію в єдиному глобальному сховищі, що суттєво оптимізує процеси маніпулювання даними та забезпечує строгу передбачуваність поведінки системи.

Архітектура клієнтської частини веб-додатка побудована за модульним принципом, де кожен структурний елемент інкапсулює специфічну бізнес-логіку. Системна архітектура поділяється на чотири базові категорії модулів:

- сервіси – спеціалізовані модулі, функціональне призначення яких зводиться до формування та маршрутизації мережевих запитів до серверної частини інформаційної системи;
- редюсери – детерміновані функції, відповідальні за безпосередню трансформацію глобального стану додатка на основі отриманих команд;

- екшени – посередницькі функції, що регламентують процеси взаємодії між компонентами візуалізації, сервісними модулями та редюсерами;
- компоненти – структурні класи, що забезпечують рендеринг графічного інтерфейсу та реєстрацію дій користувача.

Універсальний алгоритм обробки інформаційних транзакцій у розробленому програмному забезпеченні підпорядковується чітко регламентованому життєвому циклу:

1. Ініціалізація процесу шляхом взаємодії користувача з активним елементом графічного інтерфейсу.
2. Реєстрація події відповідним обробником та виклик закріпленої функції-екшена.
3. Делегування екшеном виконання мережевої операції відповідному сервісному модулю.
4. Формування та відправлення сервісом асинхронного запиту до прикладного програмного інтерфейсу серверної частини.
5. Прийом мережевого запиту контролером серверного ядра, його валідація, обробка із залученням системи управління базами даних та генерація відповіді.
6. Повернення результату виконання операції від сервера до сервісного модуля з подальшою передачею даних до функції-екшена.
7. Аналіз екшеном отриманого результату та ініціалізація виклику відповідного редюсера.
8. Модифікація глобального стану системи редюсером згідно з отриманим корисним навантаженням.
9. Реактивне оновлення компонентів графічного інтерфейсу відповідно до нового стану програмного середовища.

Візуалізація інформаційних потоків та хронології взаємодії об'єктів системи під час виконання конкретних операцій здійснюється за допомогою UML-діаграм послідовності. На рисунку 3.1 представлено діаграму послідовності, що формалізує процес вилучення товарної позиції з електронного каталогу.

Зазначена діаграма демонструє синхронну та асинхронну взаємодію п'яти ключових вузлів: обробника події, екшена, редюсера, сервісу та контролера. Життєвий цикл транзакції розпочинається з генерації виклику deleteBoxById обробником події, який передається до модулю екшена. Далі екшен ініціює виклик методу DeleteBoxById на рівні сервісу, який, у свою чергу, ретранслює відповідний мережевий запит до контролера серверної частини. Після успішного виконання операції видалення запису в базі даних, контролер повертає підтвердження у вигляді Success Message сервісному модулю. Сервіс передає це підтвердження назад до екшена. На фінальному етапі транзакції екшен генерує команду dispatch, що передається редюсеру для оновлення клієнтського стану додатка, що призводить до зникнення вилученого елемента з екрану користувача.

Отже, на рисунку 3.1 показана діаграма послідовності процесу видалення товару.

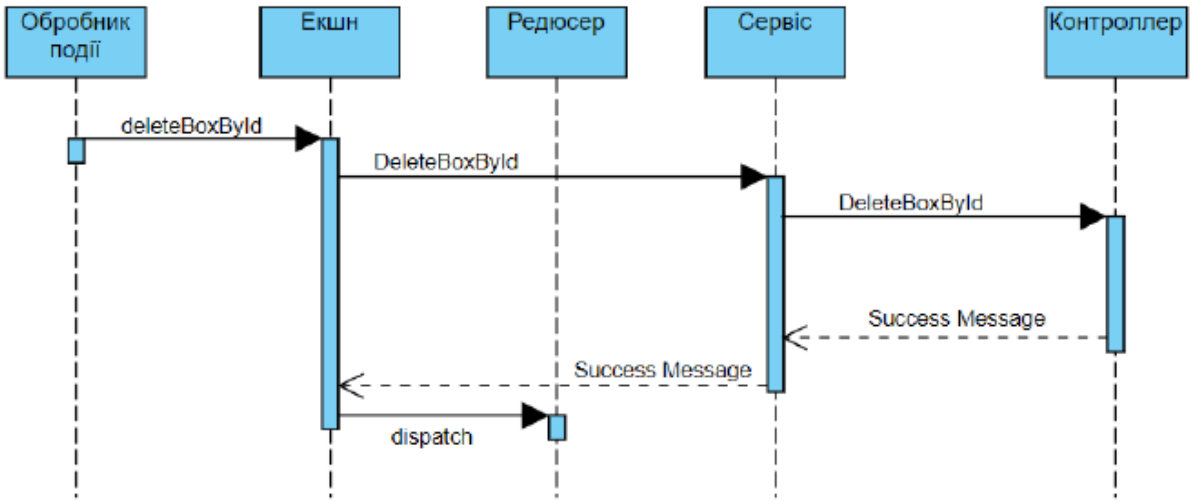


Рисунок 3.1 – Діаграма послідовності процесу видалення товару

Процес створення інформаційної бази даних ініціюється з ідентифікації ключових предметних сутностей та встановлення структурних зв'язків між ними. У контексті реляційної моделі кожна сутність проектується у вигляді окремої таблиці, де атрибути репрезентуються відповідними стовпцями. Логічна цілісність та зв'язність масивів даних забезпечується механізмами первинних та зовнішніх ключів.

Відповідно до визначених функціональних вимог системи, виокремлено базовий набір сутностей: користувач, товарна позиція, замовлення та категорійний тег. Додатковий комплекс сутностей формується як результат транзакційної взаємодії клієнта з електронним каталогом товарів. До цієї категорії належать: віртуальний кошик, призначений для тимчасового акумулювання обраних одиниць; коментар, що забезпечує збереження текстового зворотного зв'язку; рейтинг, який фіксує кількісну оцінку; та відношення, що відображає бінарну реакцію користувача у вигляді позитивного чи негативного маркування продукції.

Результат логічного проектування структури даних графічно репрезентовано на рисунку 3.2. Представлена діаграма демонструє комплексну схему реляційних відношень між ключовими вузлами зберігання інформації розроблюваної системи.

Аналіз наведеної архітектурної схеми дозволяє виокремити спеціалізований кластер таблиць із префіксом AspNet, до яких належать AspNetUsers, AspNetRoles, AspNetUserClaims та споріднені з ними структури. Зазначений інформаційний блок є стандартною апаратно-програмною генерацією підсистеми ідентифікації, що забезпечує криптографічно захищене зберігання персональних облікових даних клієнтів та комплексне управління рольовими політиками доступу. Інфраструктура зберігання

медіаданих винесена в окрему таблицю Photos, яка акумулює графічні ідентифікатори користувацьких профілів та візуальні репрезентації товарних позицій, пов'язані відповідними відношеннями з головними таблицями.

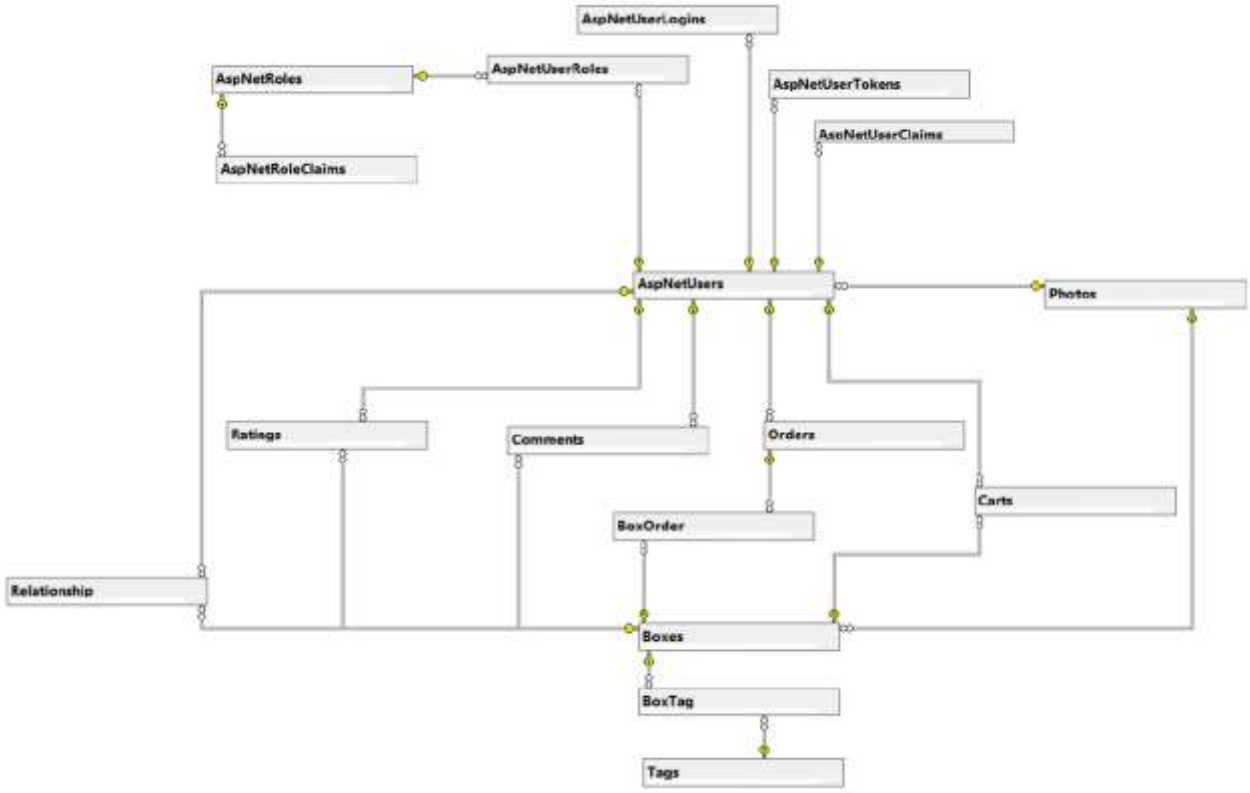


Рисунок 3.2 – Схема бази даних

Для дотримання правил нормалізації та реалізації реляційних зв'язків типу багато-до-багатьох в архітектуру інтегровано проміжні або асоціативні таблиці. Зокрема, таблиця BoxTag забезпечує гнучку категоризацію товарів шляхом поєднання ідентифікаторів з таблиць Boxes та Tags. Водночас таблиця BoxOrder фіксує точний склад кожного конкретного замовлення через агрегацію зв'язків між базою товарів та реєстром оформлених покупок. Окрім цього, на діаграмі чітко прослідковується центральна роль таблиці Boxes, яка має прямі реляційні зв'язки з таблицями відгуків Comments, оцінок Ratings та відношень Relationship. Саме ці накопичені масиви

перехресних даних формують репрезентативну вибірку для подальшого аналізу та обробки інтелектуальним модулем рекомендацій.

Програмна реалізація спроектованої схеми бази даних здійснюється з використанням технології об'єктно-реляційного відображення Entity Framework Core із застосуванням парадигми пріоритету коду. Дана методологія розробки передбачає первинне проектування об'єктно-орієнтованих класів предметної області для кожної визначеної сутності. Управління життєвим циклом підключення до бази даних, забезпечення транзакційності операцій та відстеження модифікацій делегується спеціалізованому класу контексту. Архітектура даного контекстного класу інкапсулює колекції всіх ідентифікованих сутностей, містить конструктор для ініціалізації з'єднання та автоматичного розгортання структури бази даних у разі її відсутності на сервері, а також включає перевизначені методи конфігурації для детального опису властивостей полів і реляційних обмежень.

3.2. Реалізація користувацького інтерфейсу

Життєвий цикл створення клієнтської частини веб-додатку передбачає послідовне виконання двох фундаментальних етапів. На першій стадії здійснюється графічне проектування, що включає розробку візуальних макетів та побудову інтерактивних прототипів інформаційної системи. Другий етап полягає у безпосередній програмній реалізації затверджених візуальних рішень за допомогою сучасних веб-технологій та фреймворків.

Етап графічного моделювання та прототипування інтерфейсу було реалізовано із застосуванням спеціалізованого хмарного середовища Figma. Вибір даного програмного забезпечення обґрунтовується його високою

ефективністю у сфері проектування користувацького досвіду та інтерфейсів, наявністю розширеного інструментарію для спільної роботи та оптимізованим процесом створення векторних графічних елементів. Порівняно з класичними растровими графічними редакторами, платформа Figma забезпечує значно нижчий поріг входження для розробників, пропонуючи спеціалізовані рішення для побудови адаптивних сіток, управління типографікою та створення компонентних баз дизайну, що суттєво пришвидшує процес розробки та дозволяє зосередитися на ергономіці.

На рисунку 3.3 продемонстровано загальну архітектуру робочого простору графічного редактора, а також наведено приклади спроектованих візуальних макетів сторінок веб-додатку, які виступають базовим орієнтиром та технічним завданням для подальшої програмної імплементації клієнтського інтерфейсу.

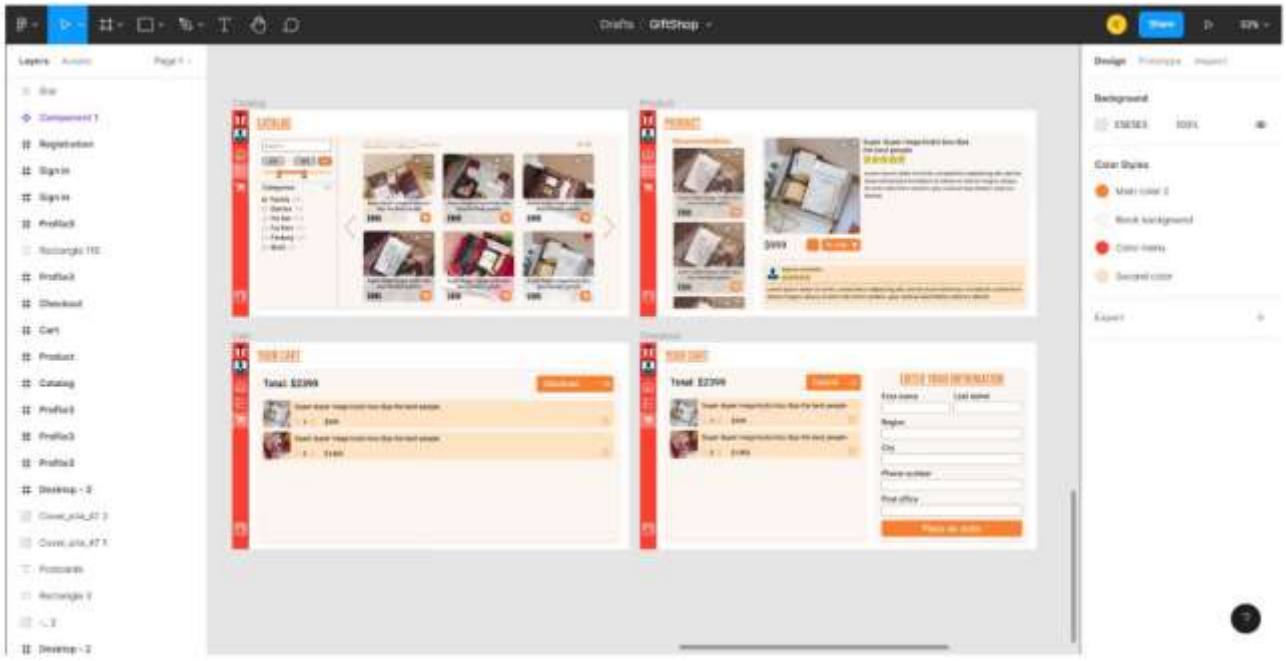


Рисунок 3.3 – Інтерфейс хмарного середовища прототипування Figma

Як зазначалося під час аналізу стеку технологій, ядром програмної реалізації клієнтської частини виступає бібліотека ReactJS. Фундаментальна парадигма даної технології базується на компонентно-орієнтованому підході, який передбачає глибоку декомпозицію складного користувацького інтерфейсу на ізольовані, логічно завершені та незалежні структурні одиниці. Кожен такий компонент відповідає за відображення окремого фрагмента графічного інтерфейсу та інкапсулює власну логіку поведінки. Системне об'єднання цих модулів утворює багаторівневу ієрархічну структуру дерева компонентів, що формує цілісний візуальний простір інформаційної системи. Раціональне архітектурне проектування компонентної бази забезпечує високий рівень їхнього повторного використання у різних сегментах веб-додатку. Це значно підвищує загальну гнучкість програмного коду, оптимізує процеси технічної підтримки та створює надійний фундамент для безперешкодного масштабування функціоналу. Подібна модульна архітектура дозволяє конструювати складні програмні комплекси шляхом композиції базових стандартизованих елементів.

З технічної точки зору, компонент у середовищі ReactJS репрезентується як функція або клас об'єктно-орієнтованого програмування, що генерує специфічний набір інструкцій для подальшого рендерингу елементів у віртуальну об'єктну модель документа. Історично та архітектурно розрізняють два основні типи компонентів. На рисунку 3.4 наведено приклад реалізації функціонального компонента, який базується на використанні чистих функцій та спеціалізованих механізмів для управління станом. Натомість рисунок 3.5 ілюструє структуру класового компонента, що використовує можливості наслідування базових класів бібліотеки та методи управління життєвим циклом відображення.

```
function Welcome(props) {
  return <h1>Привіт, {props.name}</h1>;
}
```

Рисунок 3.4 – Структура функціонального компонента

```
class Welcome extends React.Component {
  render() {
    return <h1>Привіт, {this.props.name}</h1>;
  }
}
```

Рисунок 3.5 – Структура класового компонента

Декларативний опис візуальної структури всередині компонентів здійснюється за допомогою спеціалізованого розширення синтаксису JSX. Зазначений формат не є стандартним рядковим типом даних чи чистою мовою гіпертекстової розмітки, а виступає синтаксичним доповненням над базовими функціональними викликами мови JavaScript. Використання JSX дозволяє органічно поєднувати логіку обробки даних із шаблонами їхнього візуального представлення безпосередньо в одному файлі. Це суттєво спрощує процес розробки реактивних інтерфейсів, які вимагають швидкого та динамічного оновлення контенту у відповідь на дії користувача або зміну глобального стану системи. Оформлення графічних елементів, їхнє точне позиціонування та адаптація під різні розміри екранів пристроїв реалізується шляхом інтеграції каскадних таблиць стилів.

Відповідно до обраної методології розробки, файлова архітектура клієнтської частини формується з трьох основних типів документів. Файли з розширенням .jsx містять програмну реалізацію безпосередньо візуальних компонентів інтерфейсу. Файли формату .js акумулюють допоміжну бізнес-

логіку, сервісні модулі для мережевих запитів та утиліти для обчислення масивів даних. Файли формату .css містять дескриптори стилістичного та колірного оформлення. Для забезпечення високого рівня навігації по програмному коду та запобігання його надмірній зв'язності була розроблена сувора ієрархічна структура каталогів. Усі програмні модулі логічно згруповані у відповідні директорії на основі їхньої функціональної приналежності до конкретних сторінок, секцій або системних інструментів веб-додатку. Підсумковий стан структури каталогів клієнтського застосунку графічно відображено на рисунку 3.6.

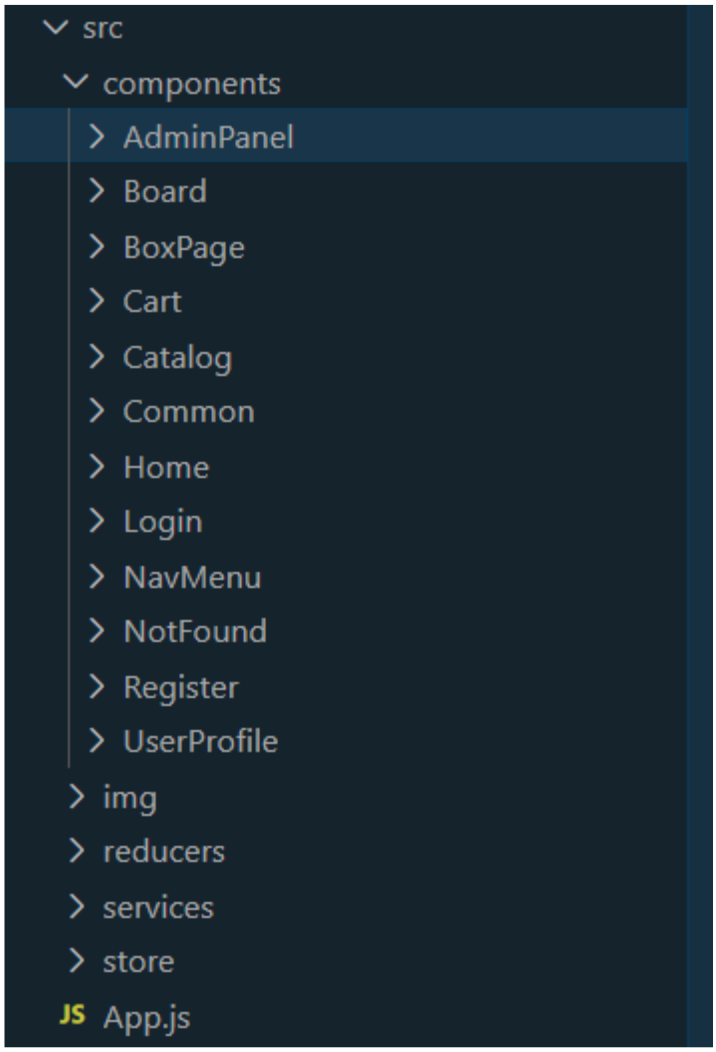


Рисунок 3.6 – Файлова структура клієнтської частини веб-додатку

3.3. Тестування роботи веб-додатку та модулів програми

Процедура створення персонального облікового запису є обов'язковою умовою для отримання користувачем повного спектру функціональних можливостей інформаційної системи. Як зазначалося під час проектування рольової моделі доступу, авторизовані суб'єкти наділяються значно ширшими повноваженнями порівняно з гостьовими відвідувачами. Ініціалізація профілю здійснюється через спеціалізовану форму реєстрації.



Рисунок 3.7 – Інтерфейс форми реєстрації нового користувача

Для успішної генерації облікового запису користувач зобов'язаний заповнити всі обов'язкові поля форми валідними даними, що відповідають встановленим критеріям безпеки (наприклад, складність пароля, унікальність електронної адреси). Завершення процедури ініціюється натисканням функціональної кнопки «Sign Up». У разі успішної валідації даних на стороні сервера відбувається автоматична переадресація на головну сторінку системи. За наявності помилок вводу або конфліктів у базі даних (наприклад,

спроба реєстрації на існуючу пошту), система генерує відповідні інформаційні повідомлення для візуалізації на екрані клієнта.

Процедура автентифікації є необхідним етапом для ідентифікації суб'єкта у системі та визначення його рівня привілеїв згідно із затвердженою матрицею контролю доступу. Автентифікація здійснюється через відповідну екранну форму. У випадку введення некоректних облікових даних система блокує доступ та виводить повідомлення про помилку авторизації.

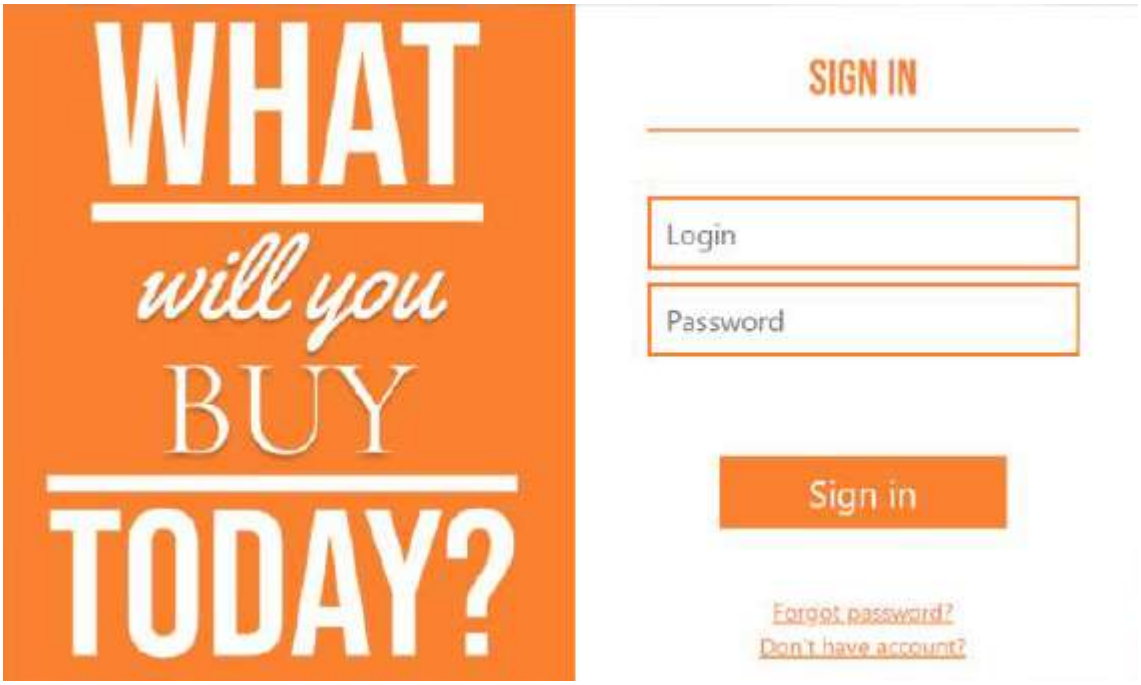


Рисунок 3.8 – Інтерфейс форми авторизації

Після успішної авторизації користувач отримує доступ до персонального кабінету – спеціалізованої сторінки профілю, що містить консолідовану особисту інформацію. Перехід до цього розділу здійснюється через клік на графічний ідентифікатор (аватар) у навігаційному меню.

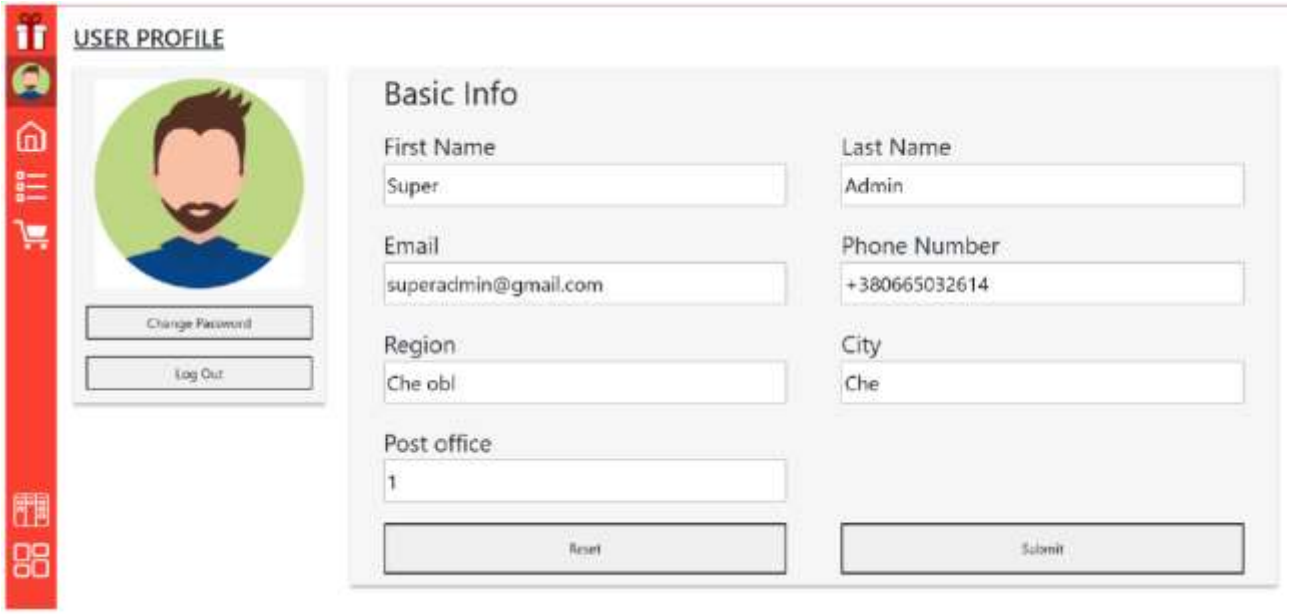


Рисунок 3.9 – Інтерфейс персонального профілю користувача

Інтерфейс профілю надає інструментарій для повноцінного управління обліковим записом: редагування персональних атрибутів, оновлення графічного ідентифікатора, модифікації параметрів безпеки (пароля) та безпечного завершення сеансу роботи (виходу із системи).

Для коригування персональних даних користувачеві необхідно активувати відповідне текстове поле на сторінці профілю, внести зміни та підтвердити операцію натисканням кнопки «Submit». Передбачено механізм відміни незбережених змін за допомогою функціональної клавіші «Reset», яка повертає значення полів до їхнього початкового стану. Оновлення візуального аватара здійснюється шляхом натискання на поточне зображення, що ініціює виклик системного діалогового вікна для вибору графічного файлу з локального сховища користувача з його подальшим асинхронним завантаженням на сервер.

Процедура зміни пароля винесена в окремий захищений сценарій. Ініціалізація процесу відбувається натисканням кнопки «Change Password»,

що призводить до рендерингу модального вікна. Алгоритм безпеки вимагає введення поточного дійсного пароля для підтвердження особи та двократного введення нового пароля. Оновлення параметрів безпеки фіксується кнопкою «Save».

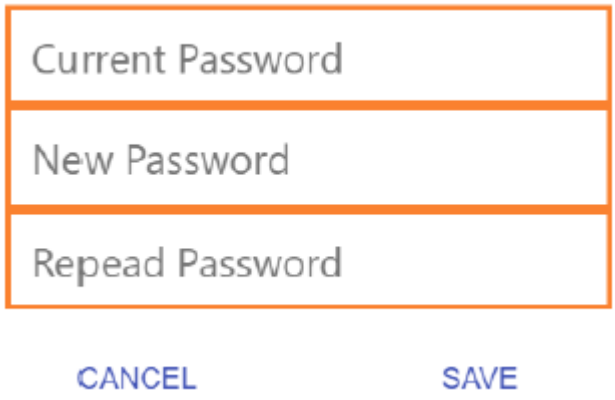


Рисунок 3.10 – Інтерфейс модального вікна зміни пароля

Завершення робочого сеансу ініціюється кнопкою «Log Out», після чого система вимагає додаткового підтвердження дії у діалоговому вікні для запобігання випадковому виходу.

Головним операційним простором веб-додатку виступає сторінка електронного каталогу, призначена для перегляду та вибору товарних позицій. Архітектура сторінки передбачає посторінковий вивід інформації з лімітом у шість номенклатурних одиниць на один екран для оптимізації швидкості завантаження. Навігація між сторінками здійснюється за допомогою графічних стрілок пагінації, а індикація поточної сторінки відносно їхньої загальної кількості розміщена у верхньому інформаційному блоці. Ліва бічна панель інтерфейсу містить комплексний інструментарій для багатовимірної фільтрації асортименту за категорійними ознаками, ціновими діапазонами та контекстним пошуком за найменуванням.

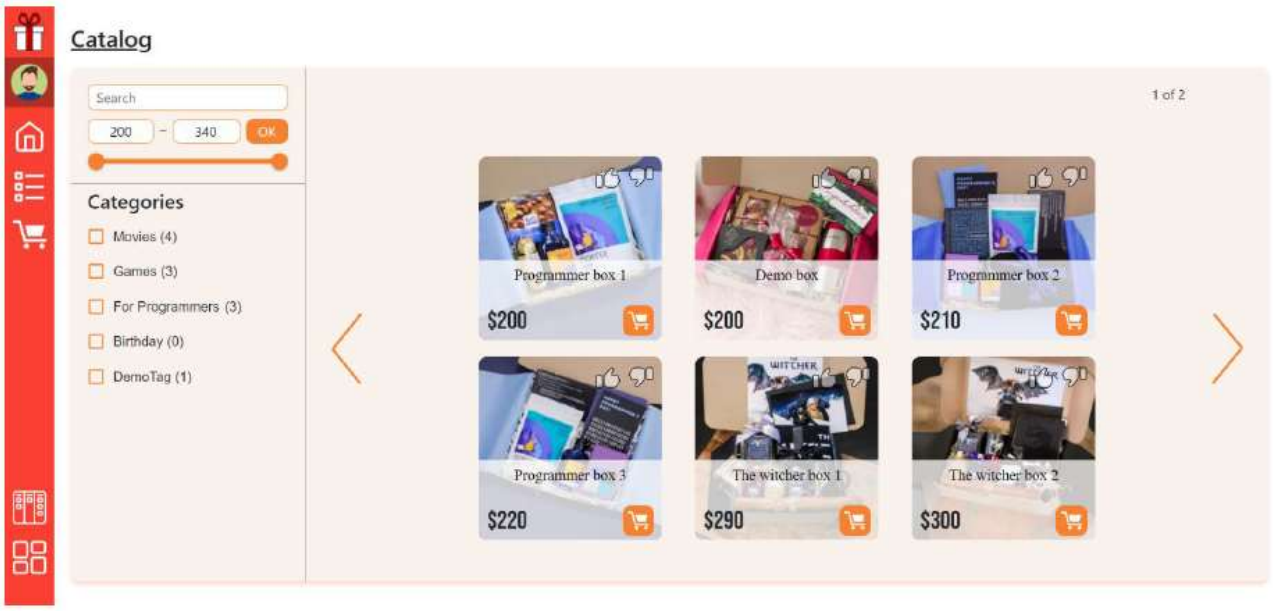


Рисунок 3.11 – Інтерфейс електронного каталогу товарів

Картка кожного товару в каталозі містить базову комерційну інформацію (назва, вартість) та інтерактивні елементи для оперативного додавання позиції до віртуального кошика або формування оціночного судження (лайк/дизлайк).

Для перегляду розширених специфікацій користувач може перейти на детальну сторінку товару шляхом натискання на його заголовок. Структура деталізованої сторінки включає вичерпний опис продукту, модуль користувацьких рецензій та блок персоналізованих рекомендацій, згенерованих інтелектуальним модулем.

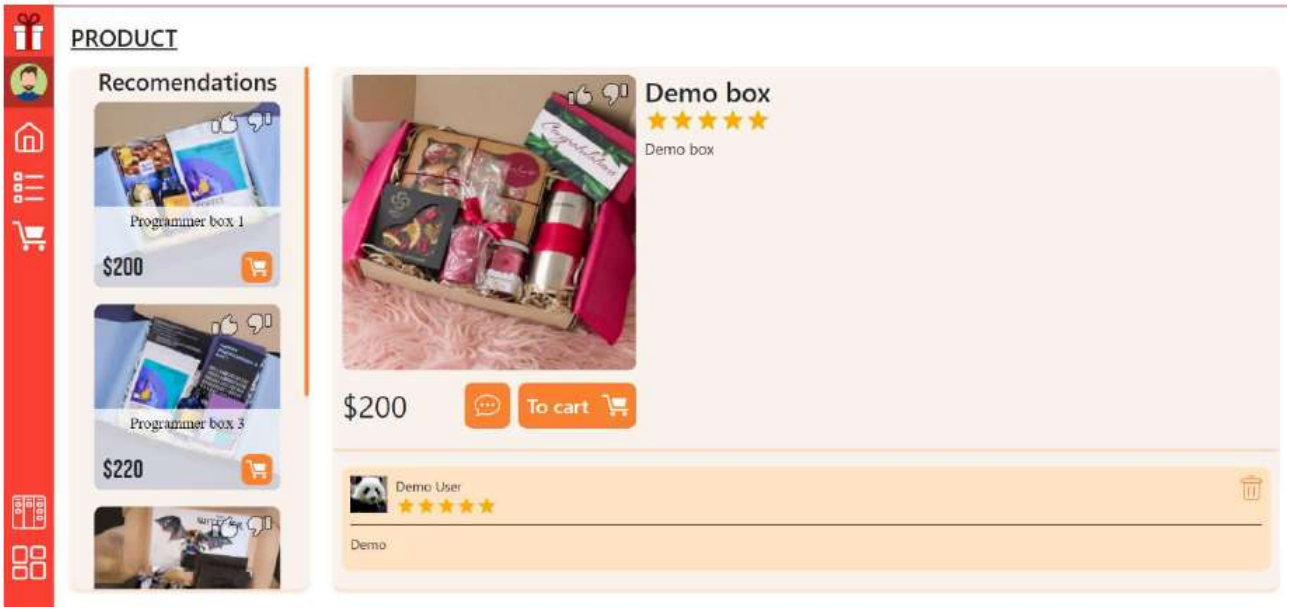


Рисунок 3.12 – Деталізована сторінка товарної позиції

Інтерфейс взаємодії з відгуками дозволяє користувачам брати участь у формуванні соціального рейтингу. Клік по кнопці написання відгуку ініціює виклик модального вікна, що містить текстове поле для введення рецензії та компонент для виставлення кількісної оцінки. Публікація здійснюється через кнопку «Submit». Авторизовані користувачі мають права на керування власним контентом, включаючи можливість видалення раніше опублікованих коментарів за допомогою відповідної іконки.



Рисунок 3.13 – Інтерфейс модального вікна публікації рецензії

Перехід до розділу управління замовленням здійснюється через іконку віртуального кошика в навігаційному меню. Ця підсистема дозволяє користувачеві здійснити фінальний аудит обраних товарів та ініціювати процедуру їх придбання.

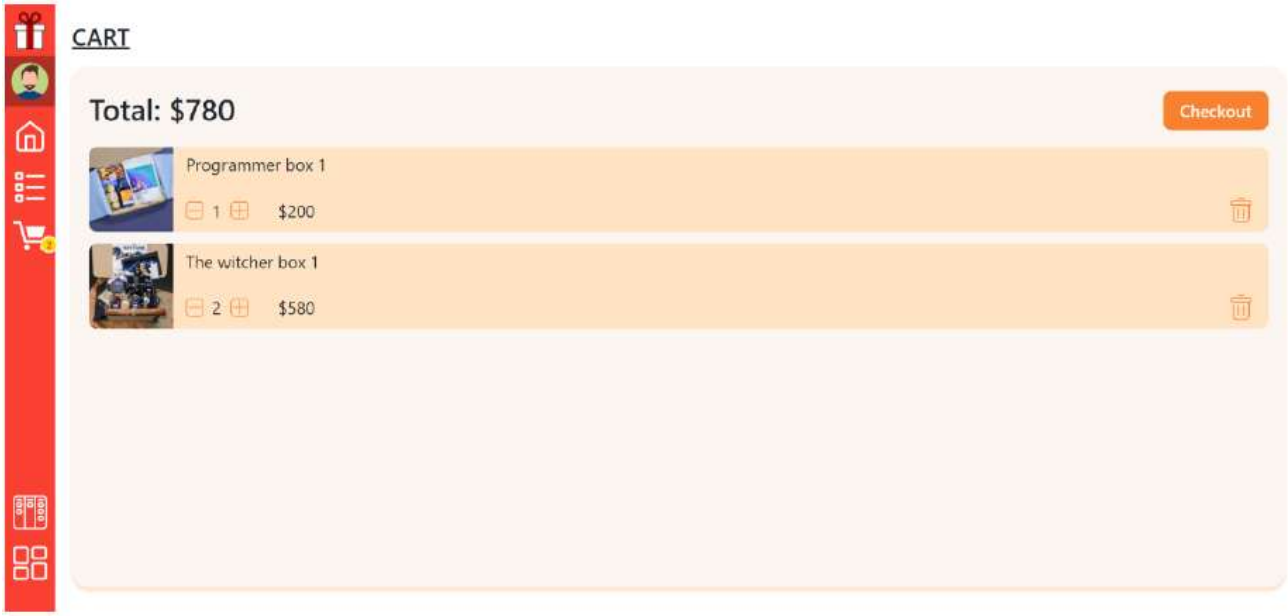


Рисунок 3.14 – Інтерфейс віртуального кошика

Функціонал кошика забезпечує можливість динамічного коригування кількості кожної доданої позиції за допомогою кнопок інкрементації та декрементації. Повне вилучення товару з переліку здійснюється натисканням іконки видалення.

Після завершення процесу комплектації кошика перехід до оформлення замовлення ініціюється кнопкою «Checkout». Ця дія викликає спеціалізовану форму збору логістичних та контактних даних.

Користувач повинен заповнити форму необхідною інформацією та підтвердити операцію кнопкою «Place an order», після чого замовлення реєструється в інформаційній системі та передається на обробку.



Рисунок 3.15 – Форма реєстрації даних для доставки

Операційне управління потоковими замовленнями здійснюється через спеціалізовану інформаційну дошку, доступ до якої регламентується рольовою політикою виключно для менеджерів та адміністраторів. Інтерфейс дошки побудований за принципом канбан-системи та розподілений на три логічні секції, що відповідають статусам обробки: «TO DO» (нові), «IN PROGRESS» (в обробці) та «DONE» (завершені).

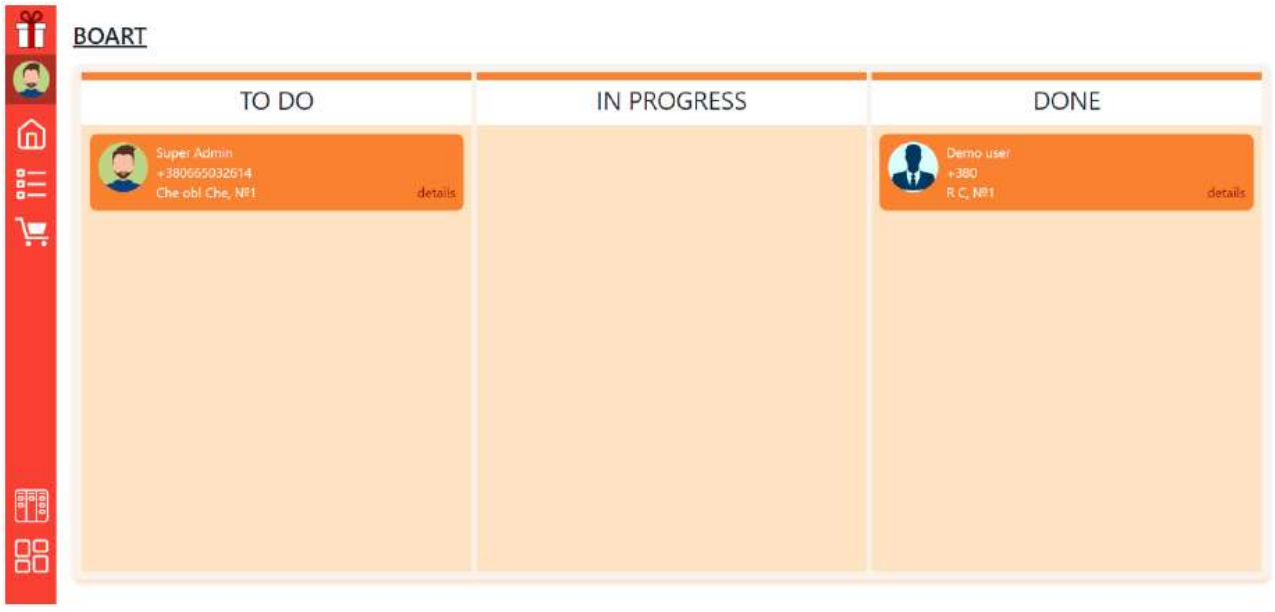


Рисунок 3.16 – Інтерфейс інформаційної дошки управління замовленнями

Кожна картка на дошці репрезентує окреме замовлення і містить базові ідентифікаційні дані клієнта (ПІБ, контактний телефон, адреса логістики). Для доступу до розширеної специфікації замовлення передбачена кнопка «details».



Рисунок 3.17 – Модальне вікно розширеної інформації про замовлення

Вказане модальне вікно надає повну деталізацію вмісту замовлення та містить функціональний елемент для зміни його поточного статус-коду в системі.

Глобальне управління платформою здійснюється через панель адміністратора, доступ до якої жорстко обмежений найвищим рівнем системних привілеїв. Інтерфейс панелі структурований на три основні функціональні модулі: управління товарним асортиментом, управління категорійною структурою та адміністрування клієнтської бази.

Модуль управління товарами надає адміністратору інструменти для перегляду реєстру продукції, створення нових записів, редагування існуючих специфікацій та вилучення позицій з бази даних.



Рисунок 3.18 – Інтерфейс модуля управління товарами в адміністративній панелі

Ініціалізація процесу додавання нового товару відбувається через кнопку «ADD NEW BOX», що викликає відповідну форму заповнення атрибутів продукту: завантаження медіа-файлу, введення найменування, встановлення цінової політики, формування текстового опису та прив'язки до категорійного дерева. Реєстрація позиції завершується натисканням «Submit».

Процедура модифікації існуючого товару ідентична процесу його створення, за винятком того, що форма попередньо заповнюється поточними значеннями з бази даних. Вилучення товарної одиниці з каталогу реалізується натисканням іконки видалення на відповідній картці.

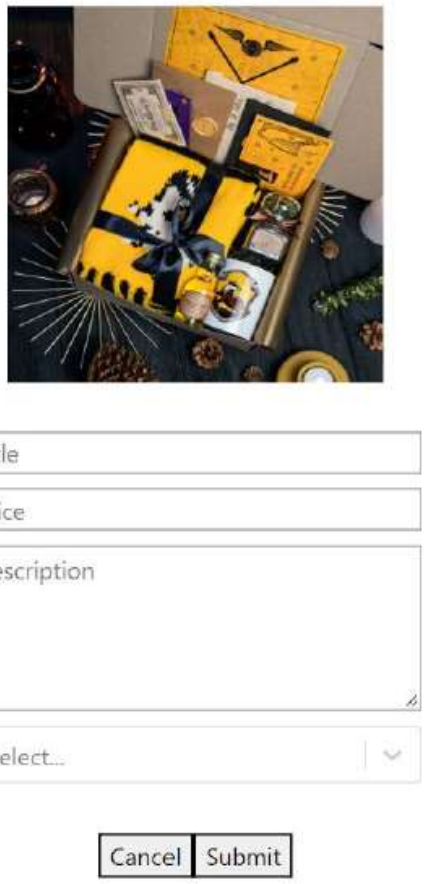


Рисунок 3.19 – Інтерфейс модального вікна створення/редагування товару

Модуль управління категоріями, представлений розділом «Tags», забезпечує повний цикл CRUD-операцій (створення, читання, оновлення, видалення) над об'єктами категорійної ієрархії.



Рисунок 3.20 – Інтерфейс модуля управління категоріями

Створення нової категорії ініціюється кнопкою «ADD NEW TAG», яка генерує базовий об'єкт із системним найменуванням. Подальше налаштування здійснюється шляхом переходу в режим редагування картки категорії. Аналогічний механізм застосовується для модифікації назв існуючих категорій, а їх вилучення проводиться через стандартну функцію видалення.

Модуль адміністрування клієнтської бази надає інструментарій для моніторингу списку зареєстрованих суб'єктів та гнучкого управління їхніми системними правами доступу.



Рисунок 3.21 – Інтерфейс модуля адміністрування клієнтської бази

Для оптимізації роботи з великими масивами даних у модулі реалізовано механізми пошукової фільтрації за іменами користувачів та сортування профілів за закріпленими за ними системними ролями. Процедура зміни рівня привілеїв користувача здійснюється безпосередньо у списку шляхом вибору необхідного значення із контекстного меню ролей.

Забезпечення надійності та безвідмовності функціонування розробленого програмного забезпечення вимагає проведення комплексного тестування. Базовим рівнем верифікації кодової бази є модульне (юніт) тестування, яке спрямоване на ізольовану перевірку коректності роботи окремих компонентів та функцій системи. Цей етап є критично важливою складовою життєвого циклу розробки, оскільки дозволяє превентивно виявляти та локалізовувати логічні помилки на ранніх стадіях написання коду.

Оскільки архітектура інформаційної системи делегує ключові процеси обробки даних серверному ядру, фокус модульного тестування зосереджено саме на верифікації бізнес-логіки бекенд-сервісів. Програмна імплементація тестових сценаріїв здійснена із застосуванням спеціалізованого фреймворку NUnit.

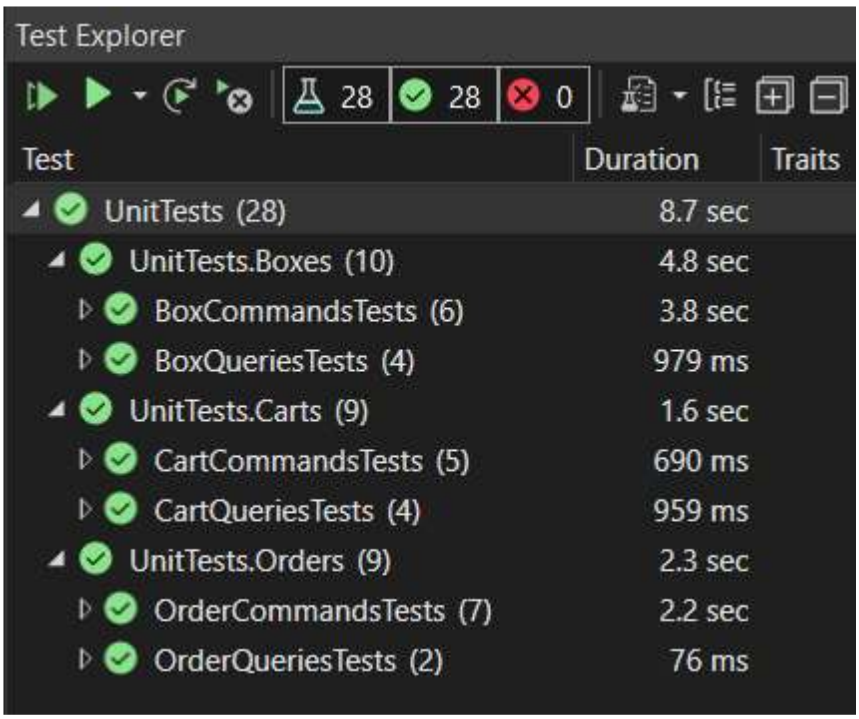


Рисунок 3.22 – Звіт про результати виконання набору модульних тестів

3.4. Програмна реалізація інтелектуального модуля рекомендацій

Програмна реалізація інтелектуального модуля рекомендацій на основі колаборативної фільтрації здійснюється з використанням засобів машинного навчання платформи ML.NET, інтегрованих у серверне середовище ASP.NET Core. Процес розробки та функціонування даного модуля поділяється на декілька послідовних етапів, що охоплюють підготовку інфраструктури даних, тренування предиктивної моделі та її безпосереднє застосування для генерації персоналізованих пропозицій у режимі реального часу.

На першому етапі відбувається формування навчальної вибірки. За допомогою технології об'єктно-реляційного відображення Entity Framework Core із реляційної бази даних Microsoft SQL Server витягується історична інформація про взаємодію користувачів із товарним асортиментом. Ця інформація включає ідентифікатори суб'єктів системи, ідентифікатори номенклатурних одиниць та відповідні кількісні міри взаємодії, сформовані на основі виставлених рейтингів. Отриманий масив неструктурованих зв'язків трансформується у спеціалізовану табличну структуру, придатну для обробки алгоритмами машинного навчання, де кожен запис репрезентує вектор взаємодії між конкретним клієнтом та товаром.

Наступним кроком є побудова обчислювального конвеєра машинного навчання та безпосереднє тренування моделі. Для програмної реалізації математичного апарату колаборативної фільтрації застосовується алгоритм матричної факторизації, базово імплементований у бібліотеці ML.NET. У процесі конфігурації обчислювального алгоритму декларуються ключові гіперпараметри, зокрема розмірність прихованого простору ознак, гранична кількість епох навчання та швидкість градієнтного спуску. Після ініціалізації налаштувань запускається процес апроксимації на підготовленому наборі історичних даних. В ході навчання алгоритм ітеративно оптимізує вагові

коефіцієнти матриць користувачів та товарів з метою мінімізації середньоквадратичної похибки між фактичними та прогнозованими значеннями рейтингів.

Після успішного завершення обчислювального циклу та валідації точності згенерована предиктивна модель серіалізується та інтегрується в архітектуру веб-додатку. Для забезпечення високої експлуатаційної продуктивності та гарантування потокобезпечності під час одночасної обробки множинних мережевих запитів використовується спеціалізований патерн ін'єкції залежностей через пул рушіїв прогнозування. Такий архітектурний підхід дозволяє серверному ядру ефективно утримувати навчену модель у кеші оперативної пам'яті та здійснювати швидкі предиктивні обчислення без надлишкових витрат ресурсів на повторну ініціалізацію об'єктів.

Процес безпосереднього надання рекомендацій ініціюється за відповідним запитом від клієнтської частини інформаційної системи. При авторизації користувача контролер прикладного програмного інтерфейсу звертається до рекомендаційного сервісу. Програмний алгоритм генерує пул товарних позицій, з якими поточний клієнт ще не мав зафіксованих взаємодій, та послідовно обчислює ймовірнісну оцінку релевантності для кожної одиниці за допомогою завантаженої моделі матричної факторизації. Результуючий масив даних проходить процедуру алгоритмічного сортування за спаданням прогнозованого рейтингу, після чого визначена кількість товарів з найвищим показником афінності формується у відповідь та передається клієнту для візуалізації у спеціалізованому інтерфейсному блоці персоналізованих пропозицій.

Для практичного підтвердження описаної архітектури нижче наведено базову програмну реалізацію інтелектуального модуля рекомендацій мовою C# з використанням бібліотеки ML.NET у середовищі ASP.NET Core.

Процес розробки розпочинається зі створення класів моделей даних, які репрезентують структуру історичних взаємодій та формат очікуваного прогнозу. Клас вхідних даних містить ідентифікатори користувача та товару, а також кількісну оцінку. Клас прогнозу визначає структуру вихідного результату алгоритму (рис. 3.23).

```
using Microsoft.ML.Data;

public class ProductRating
{
    [LoadColumn(0)]
    public float UserId { get; set; }

    [LoadColumn(1)]
    public float ProductId { get; set; }

    [LoadColumn(2)]
    public float Label { get; set; }
}

public class ProductRatingPrediction
{
    public float Label { get; set; }
    public float Score { get; set; }
}
```

Рисунок 3.23 – Клас вхідних даних

Наступним етапом є реалізація логіки побудови та тренування предиктивної моделі. Для цього розробляється сервісний клас, що використовує базовий контекст машинного навчання для конфігурації алгоритму матричної факторизації. У наведеному лістингу визначаються ключові гіперпараметри, такі як кількість ітерацій навчання та ранг апроксимації (рис. 3.24).

```
using Microsoft.ML;
using Microsoft.ML.Trainers;
using System.Collections.Generic;

public class RecommendationModelBuilder
{
    private readonly MLContext _mlContext;

    public RecommendationModelBuilder()
    {
        _mlContext = new MLContext();
    }

    public ITransformer BuildAndTrainModel(IEnumerable<ProductRating> trainingData)
    {
        IDataView trainingDataView = _mlContext.Data.LoadFromEnumerable(trainingData);

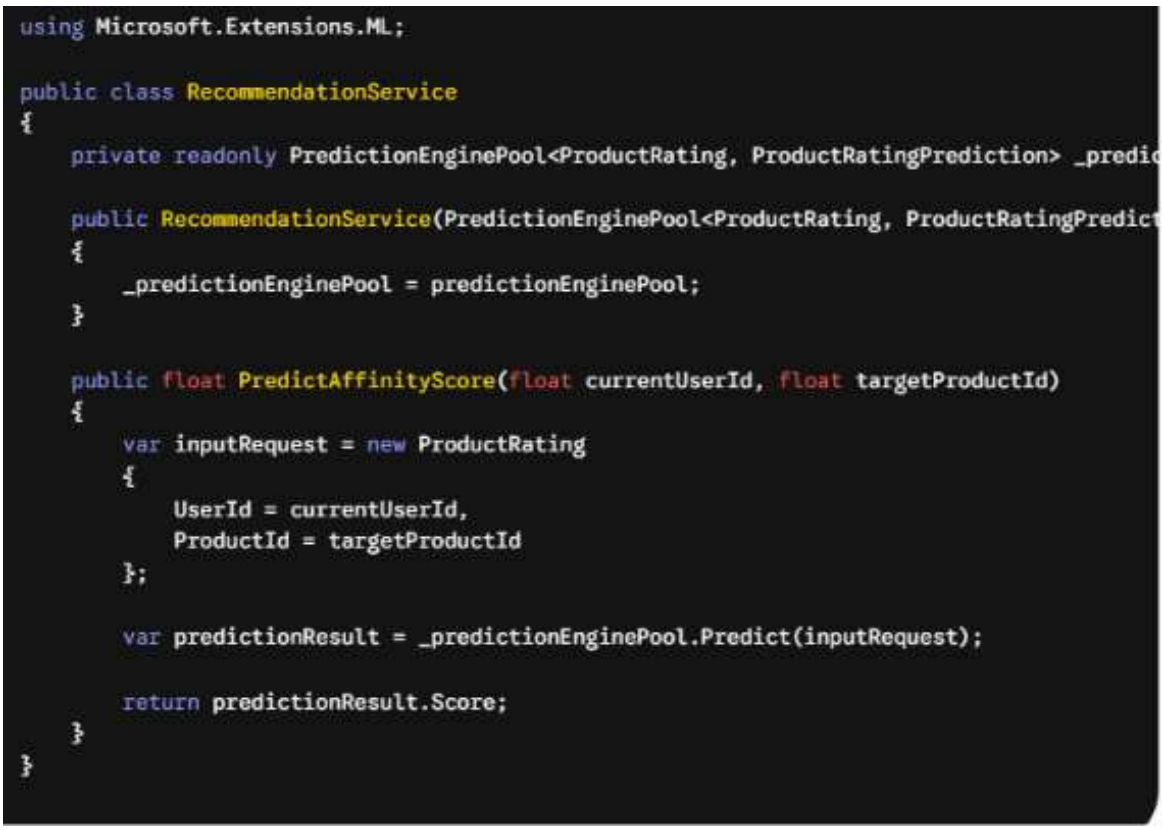
        var options = new MatrixFactorizationTrainer.Options
        {
            MatrixColumnIndexColumnName = nameof(ProductRating.UserId),
            MatrixRowIndexColumnName = nameof(ProductRating.ProductId),
            LabelColumnName = "Label",
            NumberOfIterations = 20,
            ApproximationRank = 100
        };

        var estimator = _mlContext.Recommendation().Trainers.MatrixFactorization(options);
        ITransformer trainedModel = estimator.Fit(trainingDataView);

        return trainedModel;
    }
}
```

Рисунок 3.24 – Конфігурація алгоритму матричної факторизації

Для застосування моделі у багатопотоковому веб-середовищі реалізується сервіс прогнозування. Завдяки використанню патерну впровадження залежностей та спеціалізованого пулу рушіїв прогнозування, система здатна ефективно обробляти запити без необхідності повторного завантаження моделі в оперативну пам'ять (рис. 3.25).



```
using Microsoft.Extensions.ML;

public class RecommendationService
{
    private readonly PredictionEnginePool<ProductRating, ProductRatingPrediction> _predic

    public RecommendationService(PredictionEnginePool<ProductRating, ProductRatingPredict
    {
        _predictionEnginePool = predictionEnginePool;
    }

    public float PredictAffinityScore(float currentUserId, float targetProductId)
    {
        var inputRequest = new ProductRating
        {
            UserId = currentUserId,
            ProductId = targetProductId
        };

        var predictionResult = _predictionEnginePool.Predict(inputRequest);

        return predictionResult.Score;
    }
}
```

Рисунок 3.25 – Процес застосування моделі

Представлений програмний код інкапсулює повний цикл роботи інтелектуального модуля рекомендацій. На основі обчисленого показника Score контролер інформаційної системи здійснює сортування товарного каталогу та формує фінальну вибірку персоналізованих пропозицій для подальшої візуалізації на клієнтській стороні.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно вирішено актуальне науково-практичне завдання, що полягало у проектуванні та розробці веб-додатку для інтернет-магазину з інтелектуальним модулем рекомендацій. Відповідно до поставленої мети було виконано наступний комплекс робіт.

По-перше, проведено ґрунтовний системний аналіз предметної області електронної комерції та досліджено існуючі програмні аналоги. Виявлено їхні функціональні обмеження, зокрема відсутність підсистем зворотного зв'язку та інструментів персоналізації контенту. На основі отриманих результатів сформовано специфікацію вимог до проектованої інформаційної системи та розроблено рольову модель доступу з детальним поділом повноважень між неавторизованими користувачами, зареєстрованими клієнтами, менеджерами та адміністраторами.

По-друге, здійснено комплексне архітектурне проектування системи на базі клієнт-серверної моделі з використанням принципів чистої архітектури. Це забезпечило високий рівень модульності, масштабованості та незалежності ядра бізнес-логіки від зовнішніх інфраструктурних компонентів. Спроектовано логічну та фізичну структуру реляційної бази даних, яка гарантує надійне збереження інформації про клієнтську базу, товарний асортимент, реєстр замовлень та соціальні взаємодії користувачів.

По-третє, програмно реалізовано повнофункціональний веб-додаток із застосуванням сучасного стеку технологій. Серверна частина побудована на платформі .NET із впровадженням технологій об'єктно-реляційного відображення для ефективного доступу до даних. Клієнтська частина реалізована за допомогою передових бібліотек для побудови реактивних користувацьких інтерфейсів та інструментів прогнозованого управління

глобальним станом. Графічний інтерфейс був попередньо спроектований у хмарному середовищі прототипування з дотриманням сучасних стандартів ергономіки та адаптивності.

По-четверте, спроектовано та інтегровано інтелектуальний модуль рекомендацій. Математичний апарат даного підрозділу базується на синергетичному поєднанні алгоритмів матричної факторизації для забезпечення колаборативної фільтрації та методів частотного аналізу ознак для контентно-орієнтованої фільтрації. Такий комплексний підхід дозволив забезпечити високу релевантність персоналізованих пропозицій та ефективно вирішити проблему холодного старту. Побудова, навчання та інтеграція предиктивної моделі здійснені за допомогою спеціалізованого фреймворку машинного навчання з урахуванням вимог багатопотокового середовища.

По-п'яте, проведено всебічне тестування розробленого програмного забезпечення. Впровадження методології модульного тестування забезпечило ізольовану верифікацію ключових компонентів серверної бізнес-логіки, що дозволило підтвердити високий рівень надійності, стабільності та відмовостійкості створеної інформаційної системи.

Одержані результати формують завершений програмний продукт, який має вагоме практичне значення. Розроблений веб-додаток є готовим до експлуатації рішенням, впровадження якого дозволить торговельним підприємствам суттєво оптимізувати операційні процеси, підвищити якість дистанційного обслуговування клієнтів та максимізувати комерційну ефективність за рахунок впровадження алгоритмів алгоритмічної персоналізації продажів.

СПИСОК ЛІТЕРАТУРИ

1. Freeman A. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. 9th ed. Apress, 2022. 1362 p.
2. Price M. J. C# 10 and .NET 6 – Modern Cross-Platform Development. 6th ed. Packt Publishing, 2021. 822 p.
3. Troelsen A., Japikse P. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. 11th ed. Apress, 2022. 1652 p.
4. Albahari J. C# 10 in a Nutshell: The Definitive Reference. O'Reilly Media, 2022. 1088 p.
5. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.
6. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. 560 p.
7. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2003. 560 p.
8. Smith J. Entity Framework Core in Action. 2nd ed. Manning Publications, 2021. 592 p.
9. Ben-Gan I. T-SQL Fundamentals. 3rd ed. Microsoft Press, 2016. 464 p.
10. Coronel C., Morris S. Database Systems: Design, Implementation, & Management. 13th ed. Cengage Learning, 2018. 864 p.
11. Date C. J. An Introduction to Database Systems. 8th ed. Pearson, 2003. 1008 p.
12. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. Київ: Видавнича група BHV, 2006. 384 с.
13. Грицюк Ю. І., Рак Т. Є. Інженерія програмного забезпечення: навчальний посібник. Львів: Вид-во Львівської політехніки, 2011. 400 с.
14. Заболотній В. І. Проектування інформаційних систем: навчальний посібник. Київ: Київський національний торговельно-економічний університет, 2015. 420 с.
15. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. O'Reilly Media, 2020. 336 p.

16. Wieruch R. The Road to React: Your journey to master plain yet pragmatic React.js. 2021. 228 p.
17. Flanagan D. JavaScript: The Definitive Guide. 7th ed. O'Reilly Media, 2020. 706 p.
18. Simpson K. You Don't Know JS Yet: Get Started. 2nd ed. O'Reilly Media, 2020. 143 p.
19. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. 2nd ed. Springer, 2015. 1003 p.
20. Aggarwal C. C. Recommender Systems: The Textbook. Springer, 2016. 498 p.
21. Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems. Computer. 2009. Vol. 42, № 8. P. 30–37.
22. Lops P., de Gemmis M., Semeraro G. Content-based Recommender Systems: State of the Art and Trends. Recommender Systems Handbook. Springer, 2011. P. 73–105.
23. Burkov A. The Hundred-Page Machine Learning Book. Andriy Burkov, 2019. 160 p.
24. Alpaydin E. Introduction to Machine Learning. 4th ed. MIT Press, 2020. 712 p.
25. Кравець П. О. Основи машинного навчання: навчальний посібник. Львів: Видавництво Львівської політехніки, 2020. 228 с.
26. Bress J. Machine Learning with ML.NET. Apress, 2021. 280 p.
27. Osherove R. The Art of Unit Testing: with examples in C#. 3rd ed. Manning Publications, 2021. 328 p.
28. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. Addison-Wesley, 2005. 496 p.
29. Norman D. The Design of Everyday Things: Revised and Expanded Edition. Basic Books, 2013. 368 p.
30. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. New Riders, 2014. 216 p.
31. Laudon K. C., Traver C. G. E-commerce 2021-2022: Business. Technology. Society. 16th ed. Pearson, 2021. 944 p.

32. Chaffey D., Hemphill T., Edmundson-Bird D. Digital Business and E-Commerce Management. 7th ed. Pearson, 2019. 680 p.
33. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 816 p.
34. ASP.NET Core Fundamentals. Official Microsoft Documentation. URL: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/
35. Entity Framework Core Documentation. Official Microsoft Documentation. URL: https://learn.microsoft.com/en-us/ef/core/
36. ML.NET Documentation. Official Microsoft Documentation. URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/
37. React – A JavaScript library for building user interfaces. Official Documentation. URL: https://reactjs.org/
38. Redux – A Predictable State Container for JS Apps. Official Documentation. URL: https://redux.js.org/
39. Figma Help Center & Documentation. Official Figma Documentation. URL: https://help.figma.com/
40. NUnit Documentation. Official NUnit Website. URL: https://docs.nunit.org/

ДОДАТКИ