ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтелектуальна рекомендаційна система для персоналізації підбору спортивного спорядження»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Даніла РЕКУЦ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-63
Даніла РЕКУЦ

Керівник: Наталія ТРІНТІНА
канд. техн. наук,
доц.

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рекуцу Данілі Едуардовичу

1. Тема кваліфікаційної роботи: «Інтелектуальна рекомендаційна система для персоналізації підбору спортивного спорядження»

керівник кваліфікаційної роботи Наталія ТРІНТІНА канд. техн. наук, доц.,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, відомості про алгоритми формування рекомендацій, параметри профілю користувача, характеристики та атрибути спортивного спорядження, вимога до функціоналу та точності рекомендацій, технічні вимоги до реалізації веб-додатка.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Дослідження сучасних методів побудови рекомендаційних систем для персоналізації вибору товарів.

2. Аналіз алгоритмів контентної, колобаритивної та гібридної фільтрації у веб-платформах електроної комерції.

3. Розробка гібридного алгоритму рекомендацій для персоналізованого підбору спортивного спорядження.

4. Проєктування та реалізація інформаційної рекомендаційної системи спортивного спорядження на базі веб-технологій.

5. Перелік ілюстративного матеріалу: *презентація*

1. Порівняння методів
2. Математична модель профілю користувача та товару
3. Алгоритм рекомендацій, побудований на методі машинного навчання
4. Гібрідна модель рекомендацій
5. Навчання моделі рекомендацій
6. Діаграми класів
7. Діаграми потоків даних
8. Модель бази даних
9. Екранні форми

6. Дата видачі завдання «31» жовтня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз науково-технічної літератури	31.10–04.11.25	
2	Дослідження сучасних підходів до побудови рекомендаційних систем	05.11–09.11.25	
3	Аналіз вимог до веб-системи	10.11–17.11.25	
4	Проєктування архітектури системи	18.11–27.11.25	
5	Розробка структури бази даних	28.11–04.12.25	
6	Реалізація клієнтської частини	05.12–09.12.25	
7	Інтеграція системи та тестування роботи	10.12–13.12.25	
8	Оформлення роботи: вступ, висновки, реферат	14.12–16.12.25	
9	Розробка демонстраційних матеріалів	17.12–19.12.25	

Здобувач вищої освіти _____ Даніла РЕКУЦ
(підпис)

Керівник кваліфікаційної роботи _____ Наталія ТРІНТІНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 81 стор., 27 табл., 43 рис., 38 джерел.

Мета роботи – оптимізація підходу для підбору спортивного спорядження за допомогою інтелектуальної рекомендаційної системи з урахуванням індивідуальних характеристик, параметрів та вподобань користувача з метою формування персоналізованих рекомендацій у браузерному середовищі.

Об'єкт дослідження – процес індивідуального підбору спортивного спорядження на основі індивідуальних даних користувача та його спортивних вподобань.

Предмет дослідження – методи та алгоритми побудови гібридної рекомендаційної системи на основі контентної фільтрації для персоналізованого підбору спортивного спорядження з урахуванням профілю користувача, рівня підготовки, цілей тренувань і поведінкових патернів.

У роботі використано сучасні методи та підходи, зокрема алгоритми контентної фільтрації, колаборативної фільтрації, гібридні методи рекомендацій, апарат математичної статистики, технології об'єктно-орієнтованого програмування, а також засоби веб-розробки Python з фреймоврком Flask. Основний акцент зроблено на комбінуванні алгоритмічних підходів для покращення точності рекомендацій та підвищення якості взаємодії користувача з системою.

Проведено аналіз існуючих рекомендаційних систем та підходів до персоналізованого підбору товарів. Вивчено принципи роботи систем на основі профілю користувача, контент-орієнтованих методів, методів пошуку схожих користувачів, а також технологій створення веб-додатків із клієнтсько-серверною архітектурою.

Розроблено та реалізовано веб-систему рекомендацій спортивного спорядження, що використовує гібридний підхід до формування рекомендацій,

який поєднує переваги content-based та collaborative filtering. Система дозволяє враховувати види спорту, рівень підготовки, тренувальні цілі, фізичні параметри, бюджетні обмеження та інші індивідуальні характеристики користувача.

Створено повноцінний веб-додаток на базі Flask, MySQL та HTML/CSS/JS. Реалізовано модулі керування профілем, каталогом товарів, рейтинговою системою, а також окремий модуль рекомендацій.

Результати дослідження підтверджують доцільність використання гібридних рекомендаційних алгоритмів у задачах підбору спортивного спорядження. Розроблена система забезпечує високу якість персоналізованих рекомендацій та має перспективи практичного застосування у спортивних магазинах, маркетплейсах та цифрових сервісах з обслуговування спортсменів.

КЛЮЧОВІ СЛОВА: РЕКОМЕНДАЦІЙНА СИСТЕМА, СПОРТИВНЕ СПОРЯДЖЕННЯю FLASK, MYSQL, КОНТЕНТНА ФІЛЬТРАЦІЯ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, ГІБРИДНІ АЛГОРИТМИ, ПЕРСОНАЛІЗАЦІЯ, ВЕБ-ДОДАТОК, МАШИННЕ НАВЧАННЯ.

ABSTRACT

Text part of a qualifying work for obtaining a master's degree: 125 pages, 27 tables, 43 figures, 38 sources.

Objective of the work is optimization of the approach to selecting sports equipment using an intelligent recommendation system taking into account individual characteristics, parameters and preferences of the user in order to form personalized recommendations in the browser environment.

Object of research is the process of individual selection of sports equipment based on the user's individual data and their sports preferences.

Subject of research are methods and algorithms for building a hybrid recommendation system based on content filtering for personalized selection of sports equipment taking into account the user's profile, level of training, training goals, and behavioral patterns.

The work employs modern methods and approaches, in particular content filtering algorithms, collaborative filtering, hybrid recommendation methods, mathematical statistics tools, object-oriented programming technologies, as well as Python/Flask web development tools. The main emphasis is placed on combining algorithmic approaches to improve recommendation accuracy and enhance the quality of user interaction with the system.

An analysis of existing recommendation systems and approaches to personalized product selection was conducted. The principles of user profile-based systems, content-oriented methods, similar user search methods, and technologies for creating web applications with client-server architecture were studied.

A web-based recommendation system for sports equipment was developed and implemented, using a hybrid approach to forming recommendations, which combines the advantages of content-based and collaborative filtering. The system makes it possible to take into account types of sports, level of preparation, training goals, physical parameters, budget constraints and other individual user characteristics.

A full-fledged web application was created based on Flask, MySQL and HTML/CSS/JS. Modules for profile management, product catalog, rating system, and a separate recommendations module were implemented.

The research results confirm the feasibility of using hybrid recommendation algorithms in sports equipment selection tasks. The developed system provides high-quality personalized recommendations and has prospects for practical application in sports stores, marketplaces and digital services for athletes.

KEYWORDS: RECOMMENDATION SYSTEM, SPORTS EQUIPMENT, FLASK, MYSQL, CONTENT FILTERING, COLLABORATIVE FILTERING, HYBRID ALGORITHMS, PERSONALIZATION, WEB APPLICATION, MACHINE LEARNING.

ЗМІСТ

ПЕРЕЛІК УМОВНИК ПОЗНАЧЕНЬ

API – Application Programming Interface, інтерфейс прикладного програмування, що забезпечує взаємодію між компонентами системи.

DBMS – Database Management System, система керування базами даних.

MySQL – реляційна система керування базами даних, що використовується для збереження структурованих даних користувачів та каталогу товарів.

Flask – веб-фреймворк на мові Python для створення серверної частини застосунку.

HTML – HyperText Markup Language, стандартна мова розмітки для створення веб-сторінок.

CSS – Cascading Style Sheets, каскадні таблиці стилів для оформлення сторінок.

DL – Deep Learning, глибоке навчання.

ML – Machine Learning, машинне навчання.

CRUD – Create, Read, Update, Delete, базові операції над даними в адмініструванні системи.

UI – User Interface, інтерфейс користувача.

UX – User Experience, досвід користувача.

ВСТУП

Спортивна індустрія охоплює мільйони людей, бо спорт є невід'ємною частиною культури суспільства. Для великої кількості людей спорт є стилем життя, для групи людей це є їх професійна професія [1]. Виходячи з цього, люди, які займаються спортом на професійному рівні чи на рівні хобі потребують якісної екіпіровки та спортивного спорядження для підвищення ефективності від заняття спортом та якості від тренувань.

Для якісного підбору спортивного спорядження доцільно звернутися до тренера чи до професіональних консультантів з відповідними знаннями. Але сучасні тенденції інформаційної трансформації диктують свій швидкий темп розвитку і роблять реальним створення інформаційної системи, яка на базі аналізу зможе рекомендувати людям зробити правильний вибір спортивної екіпіровки чи спорядження виходячи з їх напрямку у спорті, потреб чи кінцевої мети.

Дана кваліфікаційна робота присвячена створенню рекомендаційної системи спортивного спорядження на основі сучасних алгоритмів контентного, колаборативного та гібридного фільтрування. У процесі роботи здійснено аналіз технологій побудови вебзастосунків, зокрема використання Flask як серверного фреймворку та MySQL як системи управління базами даних. Проведено дослідження існуючих підходів до формування рекомендацій, визначено їх переваги та недоліки, а також розроблено програмний модуль, який забезпечує персоналізацію рекомендацій згідно з профілем користувача.

Особлива увага приділена структурі та взаємодії компонентів системи, реалізації алгоритмів фільтрування, а також створенню інтерфейсу, що забезпечує ефективну роботу користувача з системою. Розроблений вебзастосунок дозволяє автоматизувати процес добору спортивного спорядження, підвищити точність рекомендацій і зробити процес вибору більш швидким, зручним та інформативним.

Таким чином, завдання створення рекомендаційної системи спортивного спорядження є актуальним у контексті розвитку електронної комерції, персоналізованих сервісів і застосування сучасних методів аналізу даних.

Мета роботи: оптимізація підходу для підбору спортивного спорядження за допомогою інтелектуальної рекомендаційної системи з урахуванням індивідуальних характеристик, параметрів та вподобань користувача з метою формування персоналізованих рекомендацій у браузерному середовищі.

Об’єкт дослідження: процес індивідуального підбору спортивного спорядження на основі індивідуальних даних користувача та його спортивних вподобань.

Предмет дослідження: методи та алгоритми побудови гібридної рекомендаційної системи на основі контентної фільтрації для персоналізованого підбору спортивного спорядження з урахуванням профілю користувача, рівня підготовки, цілей тренувань і поведінкових патернів.

Для досягнення мети були поставлені наступні завдання:

1 Літературний огляд. Дослідити сучасні наукові джерела та огляди рекомендаційних систем, алгоритмів фільтрування та методів персоналізації.
2 Аналіз існуючих рішень. Розглянути підходи до формування рекомендацій у комерційних та наукових системах, визначити їх особливості та обмеження.
3 Проєктування архітектури системи. Розробити структуру вебзастосунку з використанням Flask і MySQL, визначити модель даних та ключові модулі.
4 Розробка алгоритмів рекомендацій. Реалізувати контентний, колаборативний та гібридний методи формування рекомендацій.
5 Програмна реалізація. Створити вебінтерфейс, серверну частину, модуль авторизації та роботу з базою даних.
6 Тестування системи. Перевірити точність, швидкість та якість рекомендацій, виконати аналіз ефективності алгоритмів.

Вирішення цих завдань забезпечує розробку сучасного інтелектуального програмного комплексу для підбору спортивного спорядження на основі персональних даних та машинного аналізу.

1 ОГЛЯД НАУКОВОЇ ЛІТЕРАТУРИ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Теоретичні основи предметної області

Будь-яке фізичне тренування створює для людини навантаження. Якщо мова йде про професійні тренування, які спрямовані на досягнення професійних результатів, то тут мова йде про надмінні пененавантаження організму, під час якого створюється надмірна фізична та психологічна напруга. Це створює ризик травмування та порушень в організмі. Для мінімізації цих ризиків спортсмени потребують якісного спортивного спорядження.

Люди, які займаються спортом для себе на любительському рівні також потребують якісного спортивного спорядження для того, щоб тренування і спорт приносили користі, а не робили шкоди.

За даними опитування Eurobarometer [2], лише 6% респондентів займаються спортом постійно, 32% — регулярно, тоді як 17% роблять це час від часу, а 45% взагалі не приділяють уваги фізичним вправам. Така статистика свідчить про потребу у додатковій мотивації та інструментах, що полегшують початок занять, серед яких важливу роль відіграє правильно підібране спортивне спорядження. Відповідно до статистики, можна побачити, що потенційний ринок для продажу спортивного спорядження дуже великий.

1.1.1 Поняття та класифікація спортивного спорядження

Спортивне спорядження має чітке визначення, яке сформоване в законі України про фізичну культуру і спорт [3]. Спортивне спорядження це спортивна форм, спортивний інвентар для заняття спортом та організації спортивних заходів. Простими словами спортивне спорядження це спеціальні вироби, які призначеня для занять спортом та фізичною кульурою, яке продається для використання під час заняття спортом. Спортивне спорядження треба розглядати як сукупність предметів, пристроїв для заняття спортом [4].

Якщо говорити про класифікацію базового спортивного спорядження, то до нього можна віднести спортивний одяг, спортивне взуття, аксесуари для тренувань чи заняття спортом. Далі потрібно чітко розмежувати спортивне спорядження за типом:

- Індивідуальне спортивне спорядження – це таке спорядження, яке створено для заняття спортом одним спортсменом і використовується індивідуально;
- Командне спортивне спорядження – це таке спорядження, яке розраховано для групових занять спортом і використовується групою людей.

Також для того, щоб класифікація спортивного спорядження мала багато чинникі, слід виділити класифікацію за призначенням:

- Тренувальне спортивне спорядження призначене для щоденних тренувань та заняттям спорту;
- Змагальне спортивне спорядження це спеціальне спортивне спорядження для участі у спортивних змаганнях.

Класифікація спортивного спорядження дає можливість створити фільтрацію великого обсягу різноманіття на сайтах та сервісах,щоб людям було простіше обирати його.

Також слід класифікувати спортивне спорядження за:

- Виробником;
- Споживачем.

Тобто спорядження обов’язково має класифікуватися за виробником, який виготовляє це спорядження, адже бренд виробника також дуже впливає на вибір людини при купівлі спортивного спорядження та типом кінцевого споживача.

1.1.2 Чинники, що впливають на вибір спортивного інвентарю

Виробництво спортивного інвентарю й спортивного спорядження на сьогоднішній день є найбільш розвиненою галуззю промисловості у світі. Популярність спорту як такового явища привела до збільшення кількості людей, які займаються спортом, а відповідно й підвищились продажі спортивного інвентаря та спорядження [5].

Аналіз ставлення споживачів, які є покупцями спортивного обладнання говорить, що для них важливе значення під час вибору спортивного інвентаря відіграють наступні чинники:

- Професіоналізм персоналу;
- Якість продукції;
- Функціональні характеристики;
- Вартість та співвідношення ціни та якості;
- Репутація бренду;
- Дизайн та естетичний вигляд;
- Додаткові сервіси;
- Рекомендації та відгуки інших користувачів.

Покупці намагаються обирати якісне спортивне спорядження. Таке спорядження служить довше, воно зручніше і якісніше та гарантує стабільні результати. Чим якісніше спортивне спорядження, тим рідше потрібно його замінювати. Якісне спортивне спорядження забезпечує комфортне та безпечне заняття спортом, які мінімізує можливі травми.

Вищезазначені відомості дають нам можливість зрозуміти, що вибір спортивного спорядження та інвентарю є складним багатофакторним процесом, на вибір впливають багато чинників. Для професійних спортсменів та для любителів важливо розуміти ці чинники та робити вірний вибір [6]. Слід звертати увагу на наступні чинники:

- Антропометричні характеристики користувача
- Рівень фізичної підготовки та спортивна кваліфікація
- Біомеханічні особливості та техніка виконання рухів
- Специфіка виду спорту та тренувальних завдань
- Умови експлуатації та навколишнє середовище
- Економічні фактори та бюджетні обмеження
- Ергономічність та комфорт використання
- Безпека та відповідність стандартам

– Довговічність та технічне обслуговування
– Естетичні та психологічні аспекти

Головною метою дотримання цих чинників є вибір правильного спортивного обладнання, яке безпосередньо впливає на результативність тренувань, комфорт та безпеку під час заняття спортом [7].

.3 Особливості підбору спорядження для різних видів спорту

1.1

Підбір спортивного спорядження для тренувань у значній мірі залежить від виду спорту, яким займається людина.. Відповідно, якщо брати змагальне спортивне спорядження, то потрібно його класифікувати за видами спорту,щоб розуміти цю класифікацію (див. рис. 1.1).

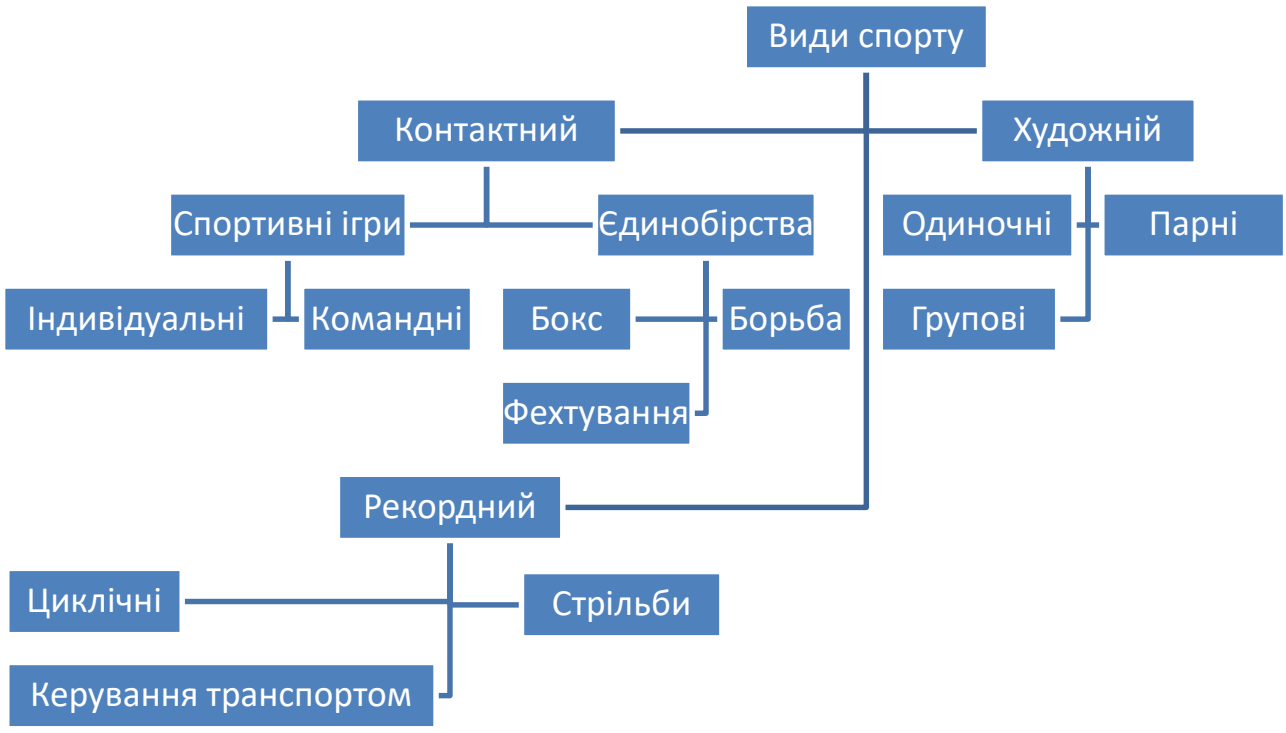


Рис. 1.1 Класифікація видів спорту та різноманіття спортивного спорядження

Такий інвентар ставить на меті забезпечити ефективність, безпеку та комфорт під час заняття спортом. Для того, щоб зрозуміти, наскільки широкий спектр

пропонуємих спортивних споряджень, необхідно оглянути популярні види спорту та створити список необхідних речей для заняття цим видом спорта.

Для бігу правильно підібране взуття [8] з урахуванням амортизації та підтримки стопи відіграє ключову роль. Бо взуття для бігуна як шини для машини. Також бігуну потрібно мати легкий та дихаючий одяг для тривалих тренувань (див. табл. 1.1).

Таблиця 1.1

Основне спорядження для бігу

Категорія	Приклад спорядження	Примітка
Взуття	Бігові кросівки з амортизацією	Для довгих дистанцій, тренувань і змагань
Одяг	Спортивні шорти, футболки, легінси	Легкий та дихаючий матеріал
Аксесуари	Годинник GPS, пульсометр	Для моніторингу результатів
Захист	Компресійні гетри, капи	Зниження ризику травм

Для людей, які займаються плаванням, потрібен підбір спортивного інвентаря та спорядження згідно їх видом спорту. Необхідними аксесуарами є купальники чи плавки, окялри для плавання та шапки для плавання [9] (див. табл. 1.2).

Таблиця 1.2

Основне спорядження для плавання

Категорія	Приклад спорядження	Примітка
Одяг	Купальник, шапочка	Аеродинамічний матеріал
Окуляри	Плавальні окуляри	Захист очей і покращення видимості
Тренувальні аксесуари	Ласти, дошки, буї	Для поліпшення техніки та сили

Для людей, які займаються фітнесом [10] чи силовими тренування у тренажерному залі потрібен зручний одяг, взуття, гантелі, штанги, тренажери і тому подібне (див. табл. 1.3).

Таблиця 1.3

Основне спорядження для фітнесу

Категорія	Приклад спорядження	Примітка
Одяг	Легкі футболки, легінси	Свобода рухів
Взуття	Кросівки для тренувань	Підтримка стопи
Обладнання	Гантелі, штанги, еспандери	Для силових вправ
Аксесуари	Килимки, фітнес-гаджети	Комфорт та безпека

Для людей, які займаються командними видами спорту, такими як футбол, баскетбол [11] чи волейбол є ряд свої особливостей у підборі спортивного спорядження (див. табл. 1.4).

Таблиця 1.4

Основне спорядження для командних видів спорту

Категорія	Приклад спорядження	Примітка
Одяг	Форма, шорти, футболки	Комфорт та свобода рухів
Взуття	Кросівки для залу/трави	Відповідно до покриття
Захист	Наколінники, щитки	Зниження ризику травм
Інвентар	М’ячі, тренувальні конуси	Для тренувань та ігор

Для людей, які займають велоспортом, зовсім інший набір потреб у спорядженні. Вони потребують велосипедів та супровідних спортивних споряджень. Також спортсменам, які займаються велоспортом потрібний одяг з амортизацією та дихаючими матеріалами. Також потрібно захисне спорядження для уникнення травмувань (див. табл. 1.5).

Таблиця 1.5

Основне спорядження для велоспорту

Категорія	Приклад спорядження	Примітка
Велосипед	Шосейний/гірський	Вибір залежить від дисципліни
Одяг	Велошорти, майки	Комфорт та аеродинаміка
Захист	Шолом, рукавички, наколінники	Безпека під час поїздок
Аксесуари	Фари, насос, фляги	Додатковий комфорт та безпека

Формування класифікації спортивного спорядження за видами спорту дало змогу зрозуміти, що в інформаційній системі, яка розробляється доцільно робити класифікацію та фільтрацію за видам спорту та іншими факторами.

1.2 Аналіз існуючих інформаційних систем рекомендацій

Сучасні системі,які вміють сторювати рекомендації стали невід’ємною частиною життя суспільства. Такі системи впроваджені скрізь, від перегляду відео на YouTube та Netflix, прослуховування музики на Spotівн до маркетплейсів,які пропанують товари на основі профіля користувача та історії переглядів товарів [12].

Тобто рекомендаційна система може рекомендувати користувачу послугу чи продукт. Щоб така система могла рекомендувати релевантний варіант, їй потрібно мати повну інформацію про користувача та його вподобання.

1.2.1 Огляд сучасних рекомендаційних систем

На сьогоднішній день можна виділити ряд рекомендаційних систем, основна суть яких однакова, пропанувати користувачу релевантний контент чи товар [13], проте системі відрізняються своїми алгоритмами та методами реалізації.

Розрізняють контекстні рекомендаційні системи, системи колаборативної фільтрації, гібридні системи та системи на основі знань (див. рис. 1.2).

Рис. 1.2 Види сучасних рекомендаційних систем

Контентна та колаборативна моделі рекомендацій часто об’єднується у гібридну [14], щоб збалансувати свої недоліки та переваги. Це дозволить рекомендувати і за інтересами користувача і за схожістю попередньо переглянутих товарів. Так модель реалізується через побудову комбінової матриці схожості (формула 1.1).

$$S = \alpha S_{CF} + (1 - \alpha) S_{CBF} \tag{1.1}$$

де $\alpha -$ ваговий коефіцієнт, $0 \leq \alpha \leq 1$.

Контентні рекомендаційні системи формують рекомендації на основі характеристик товарів та профілю користувача. Колаборативна фільтрація [15] базується на принципі схожості користувачів або товарів. Гібридні системи

поєднують декілька підходів для досягнення кращої точності та подолання обмежень окремих методів. Системи на основі знань використовують експертні правила та обмеження для формування рекомендацій, особливо корисні для складних або специфічних товарів, де потрібна консультація експерта.

1.2.2 Порівняльний аналіз відомих платформ

Для розуміння того, як рекомендаційні системи втілені в реальні успішні продукти, варто переглянути онлайн платформи по продажу спортивного спорядження з впровадженими системами рекомендацій.

Одним з лідерів ринку у галузі онлайн продажів є компанія Amazon, яка запустила напрямок Amazon Sports & Outdoor [16]. На даному сайті можна купити спортивне спорядження (див. рис. 1.3).

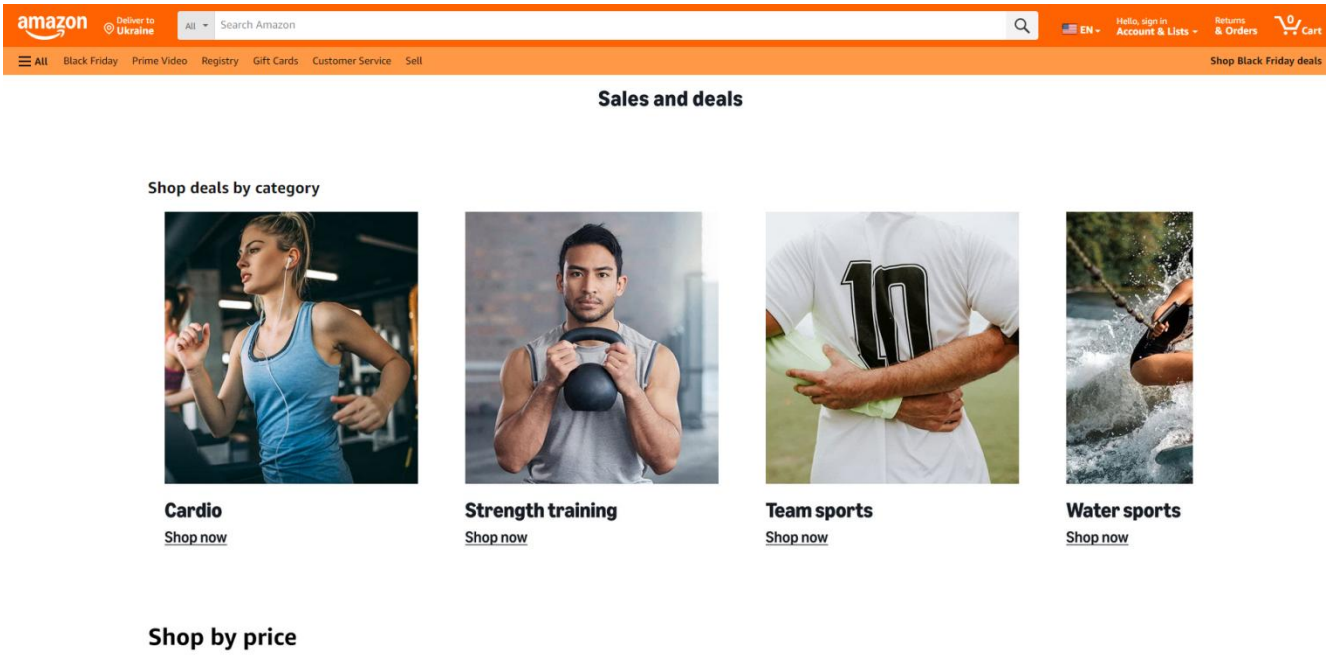


Рис. 1.3 Інтерфейс сайту Amazon

В даному проекті реалізовано складну гібридну систему рекомендацій, яка враховує історію перегляда користувача, історію його покупок, історію оцінок користувача різним товарам та кошик покупок, в якому система дивиться, які типи покупок були зроблені (див. рис. 1.4).

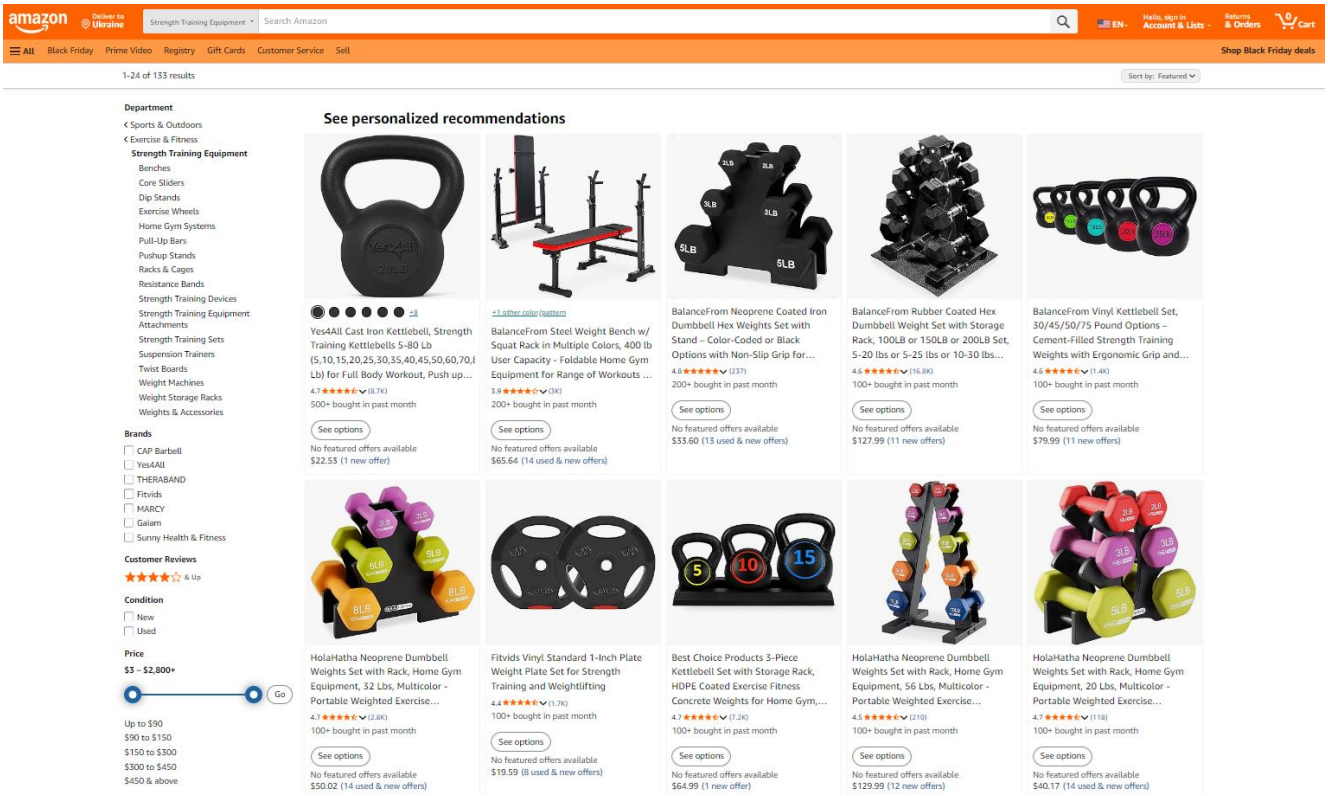


Рис. 1.4 Перегляд прикладу рекомендації від Amazon Sports & Outdoor

Система знизу каталогу відображає для користувача блок рекомендаційного контенту та персоналізовані підбірки на основі попередніх покупок. Amazon застосовує алгоритми машинного навчання для постійного удосконалення точності рекомендацій.

Зрозуміло,що Amazon це гігант з майже безмежними можливостями. Тому доцільно розглянути ще як альтернативу більш простий маркетплейс спортивного спорядження з інтеграцією системи рекомендацій. Таким є сайт Dick`s Sporting Goods [17]. Цей сайт надає базові рекомендації на основі категорій товарів, які переглядав користувач, тобто рекомендує товар зі суміжної або такої ж категорії. На сайті можна використовувати інструмент пошуку та фільтрації (див. рис. 1.5).

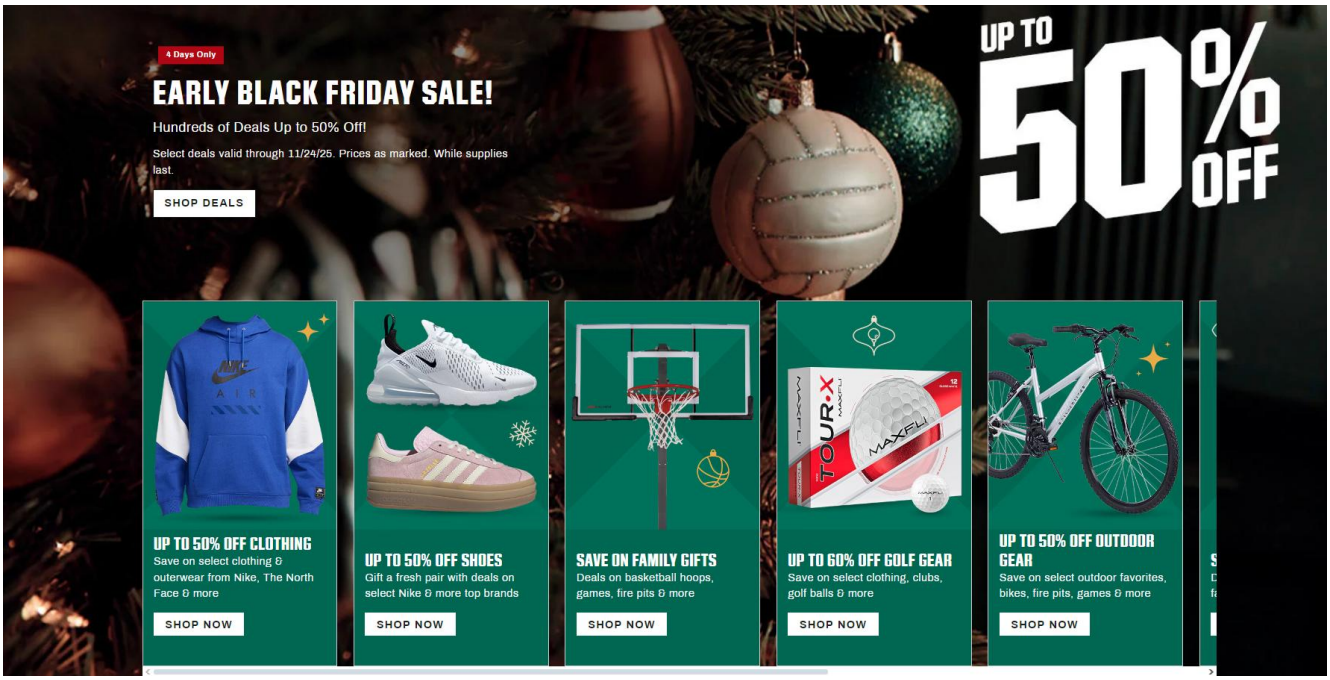


Рис. 1.5 Інтерфейс маркетплейсу спортивних товарів Dick's Sporting Goods

Таким чином сайти намагаються, щоб користувачі залишалися на ньому побільше. Це збільшує шанси на продаж та взаємодію користувача із сайтом.

1.2.3 Визначення переваг та недоліків існуючих рішень

Проаналізувавши готові рішення маркетплейсів, які впроваджують на своєму ресурсі систему рекомендацій, можна сформувати слабкі і сильні сторони для того, щоб адаптувати їх під свій проект. Звичайно, що ці великі комерційні маркетплейси мають велику кількість переваг. Домінуюча перевага полягає в наявності у них великих дата сетів даних про користувачів та їх поведінку, що допомагає їх моделям рекомендацій прохожити більш якісне навчання для підвищення точності рекомендацій. Дуже зручна штука. Як на мій погляд це інтеграція з історією покупок та переглядів. Це може забезпечити чітке попадання в інтереси клієнта без необхідності заповнення додаткової інформації. Також було виявлено, що системи намагаються рекомендувати допоміжні товари, що є супутніми до тих, які клієнт додав в кошик. Це дуже перспективне рішення, яке збільшить кількість продажів.

Проте, на мій погляд, у системі є ряд недоліків. На сайтах не врахована специфічність спортивних характеристик користувача. Можна було б брати інформацію про рівень фізичної підготовки та мету від заняття спортом. Тобто часто рекомендаційна модель базується на отриманні комерційної вигоди, а не на реальних потребах спортсмена.

Недостатня прозорість рекомендацій не дозволяє користувачам зрозуміти, чому саме було запропоновано той чи інший товар. Відсутність комплексного підходу до підбору спорядження, коли система могла б рекомендувати повний набір екіпірування для конкретного виду спорту та рівня підготовки.

1.2.4 Наявні проблеми, які потребують вирішення

Головною проблемою, яку потрібно вирішувати в першу чергу є відсутність спортивно-орієнтованого профілювання. Розглянуті системи не збирають інформацію про заняття спортом користувача, його досягнення та цілі. Саме це може значно збільшити конверсію від продажів. Це призводить до поверхневих рекомендацій, які не враховують реальні потреби спортсмена різного рівня.

Також потрібно вирішувати проблему розмежування між приватною інформацією та балансом персоналізації. Більшість систем не пояснюють, чому було рекомендовано конкретний товар, що знижує довіру користувачів та ускладнює прийняття рішення про покупку. Пояснювальні рекомендації можуть значно підвищити конверсію.

Зрозуміло,що наявність особистого спортивного профілю на сайті, який продає спортивний інвентар та спорядження могло б вирішити проблему актуальності рекомендацій. Спортивні потреби покупця будуть постійно змінюватися. І те, шо було цікаво рік тому, вже не буде актуально сьогодні [18]. Потрібно підвищувати взаємодію користувача із модулем заповнення профілю спортсмена.

1.3 Стан наукових досліджень у сфері рекомендаційних алгоритмів

Питання огляду рекомендаційних систем зазвичай розглядають на наукових конференціях в розділах, які присвячені машинному навчанні або аналізу BigData. Питання рекомендаційних алгоритмів активно обговорюється та перебуває у стані постійного прогресу .

1.3.1 Методи машинного навчання в рекомендаційних системах

Рекомендаційна система, яка працює на основі штучного інтелекту – це і є алгоритмом машинного навчання. Такий алгоритм навчається ранжувати або оцінювати товари та користувачів. Такий алгоритм робляється длятого, шоб передбачити вподобання користувача і дати йому потрібний товар або контент.

Визначення якісних рекомендації допомогає оцінити ефективність побудованої рекомендаційної системи [19]. Тут треба розглядати точність рекомендацій як частку правильних рекомендацій відносно інтересам користувача.

Методи глибокого навчання [20] революціонізували підходи до побудови рекомендаційних систем. Нейронні мережі здатні моделювати складні нелінійні залежності між характеристиками користувачів та товарів. Автокодувальники використовуються для зменшення розмірності та вилучення значущих ознак з великих обсягів даних. Рекурентні нейронні мережі та LSTM-моделі застосовуються для аналізу послідовностей дій користувачів та передбачення наступних взаємодій з урахуванням темпорального контексту.

Класичні методи колаборативної фільтрації залишаються фундаментом багатьох рекомендаційних систем. Метод найближчих сусідів на основі користувачів або товарів використовує міри схожості, такі як косинусна відстань або кореляція Пірсона, для знаходження подібних об'єктів. Матрична факторизація, зокрема метод сингулярного розкладу, дозволяє виявляти приховані латентні фактори у матриці взаємодій користувач-товар та ефективно вирішувати проблему розрідженості даних.

Контекстно-залежні методи враховують додаткові фактори, що впливають на вподобання користувачів. Це включає часовий контекст, географічне розташування, пристрій доступу та соціальні зв'язки. Тензорна факторизація розширює традиційну матричну факторизацію на багатовимірні структури даних, що дозволяє інтегрувати контекстуальну інформацію безпосередньо в модель рекомендацій (див. рис. 1.6).

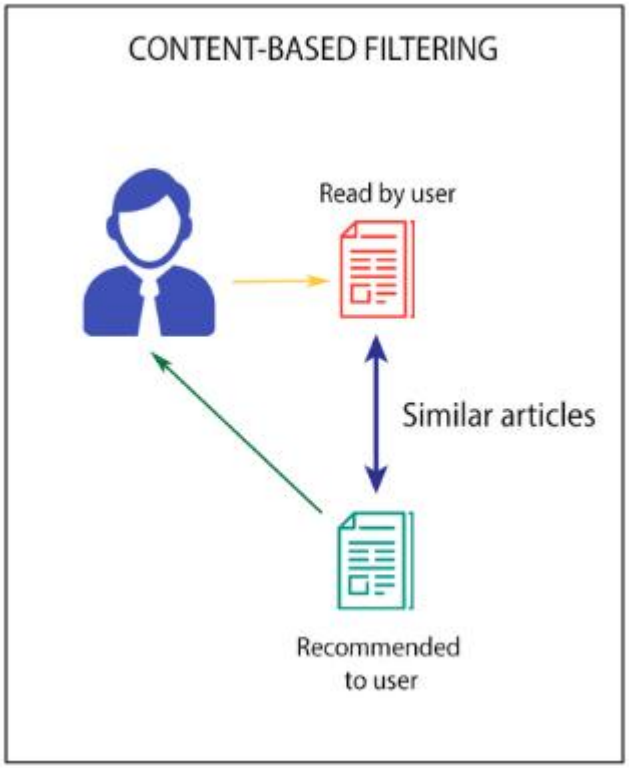


Рис. 1.6 Схема роботи контекстно-залежних методів

Гібридні підходи [21] комбінують переваги різних методів для досягнення кращих результатів. Ансамблеві методи об'єднують прогнози декількох базових алгоритмів, що підвищує надійність та точність рекомендацій. Каскадні гібридні системи послідовно застосовують різні методи, використовуючи результати попереднього етапу як вхідні дані для наступного.

Методи обробки природної мови застосовуються для аналізу текстових описів товарів, відгуків користувачів та метаданих. Word2Vec та інші методи векторного представлення слів дозволяють виявляти семантичну схожість між

товарами на основі їхніх описів. Аналіз тональності відгуків допомагає краще зрозуміти ставлення користувачів до товарів.

1.3.2 Праці українських і зарубіжних дослідників

Формування сучасних рекомендаційних систем, зокрема тих, що застосовуються для підбору спортивного спорядження, спирається на значний доробок досліджень у сфері машинного навчання. Фундаментальну теоретичну основу становлять праці C. C. Aggarwal [22] та V. Barnett і T. Lewis [23], у яких викладено статистичні, робастні й евристичні принципи аналізу даних. Ці підходи стали базою для подальшого розвитку алгоритмів класифікації, кластеризації та методів пошуку подібності, що широко застосовуються у персоналізованих рекомендаціях.

Подальшу систематизацію методів забезпечили огляди V. Chandola, A. Banerjee та V. Kumar [24], V. Hodge і J. Austin [25], а також M. A. F. Pimentel із колегами [26]. У них було запропоновано класифікацію алгоритмів за типом даних, способом навчання та структурою аномалій, що стало важливим підґрунтям для побудови адаптивних моделей поведінки користувачів у цифрових сервісах. Емпіричну перевірку цих класифікацій здійснили M. Goldstein та S. Uchida [27], демонструючи практичні можливості алгоритмів у задачах рекомендацій.

Обробка часових залежностей, важлива для аналізу спортивної активності користувачів, ґрунтується на архітектурі LSTM, представленій S. Hochreiter та J. Schmidhuber [28]. Узагальнення глибинних методів у контексті аномалій, представлене в огляді G. Pang, C. Shen, L. Cao й A. Van Den Hengel [29], також має прямий стосунок до рекомендаційних систем, що передбачають індивідуальні потреби спортсменів.

Проведені дослідження Артура Семюеля [30] показали, що машинне навчання може бути використане для надання комп’ютерам можливості навчатися без явного вказування алгоритму. Тобто можна використовувати машинне навчання для того, щоб комп’ютер міг ефективніше аналізувати дані.

Як можна побачити, велика кількість наукових праць публікується, присвячена методам машинного навчання. І це є чудово, бо дана галузь є перспективною та може перевернути все уявлення про обробку даних у світі.

1.3.3 Недостатньо досліджені аспекти, які розкриватиме автор

Хоч на сьогоднішній день і опубліковано велику кількість наукових праць, присвячених методам машинного навчання, практичні аспекти застосування та есперементні частини цього питання потребують більш щирокого розповсюдження. Залишається низка моментів, які потребують подальшого дослідження та роботи, особисто в контекті специфічних галузей.

Бальша кількість подібних робіт фокусуються на загальних методах персоналізації, яка залишається недостатньо вивченою.

Прозорість та зрозумілість рекомендацій є критично важливою для довіри користувачів, особливо коли йдеться про спортивне спорядження, яке впливає на безпеку та ефективність тренувань. Існуючі підходи до генерації пояснень часто є занадто загальними. У роботі буде досліджено методи генерації спортивно-специфічних пояснень, які враховують технічні характеристики спорядження та їхню відповідність потребам користувача.

Існуючі метрики оцінки якості прогнозування чи рекомендації треба перевіряти застосовуючи їх в різних галузях і дивитися, наскільки ці методи адаптивні та спроможні працювати з різними наборами даних в різних умовах і контекстах.

2 МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Обґрунтування вибору методів дослідження

Розробка ефективної системи машинного навчання, яка зможе навчатися та рекомендувати спортивні товари клієнтам має бути побудована з урахуванням комплексного підходу. Мають бути поєднані різноманітні методи дослідження. Галузь є специфічною, тут треба враховувати спортивні аспекти питання,характер оброблюваних даних.

2.1.1 Методи збору та аналізу даних

Якщо використати в системі метод структурованих анкет [31], то можна отримати точну та специфічну інформацію про вид спорта, поточний стан фізичної підготовки та цілі від заняття спортом користувача. Замість того,щоб неявно збирати всі інформацію про користувача, доцільно прямо запитати в нього про заповнення анкети, яка допоможе ідентифікувати його приналежність до спорту та спортивний рівень підготовки. Це не потребує тривалого часу для вивчення поведінки користувача і дає максимально ефективні знання системі. Таким чином виконується структурне профілювання, яке дозволяє отримати всю необхідну інформацію для формування рекомендацій стосовно спортивного спорядження, яке рекомендувати покупцю. Анкета має включати в себе роширені властивості, які описують параметри людини, яким видом спорту займається, рівень підготовки, бюджетні преференції.

Для спортивного спорядження доцільно застосувати принцип категоризації та атрибутації товарів. Розбиття товарі по категоріям та пропанування клієнту товарів з тієї категорії, де він щойно був є максимально ефективним. Таким чином зужується коло представлення рекомендації стосовно категорії товару. Додавання атрибутів до товару, якщо товар відноситься до категорій товарів для схуднення, то доцільно рекомендувати товари з такими самими атрибутами для схуднення.

Очевидно, що людина шукає можливості скинути зайву вагу. Доцільно для спортивного спорядження застосвувати багаторівневу структуру категоризації, де кожна категорія має свою під категорію. Гарною практикою є додавання атрибутів не лише для товарів, а й для категорій. Наприклад, категорія може включати інформацію про вид спорту, рівень підготовки фізичний людини та цінову категорію. Якщо кожен товар описати набором структурованих атрибутів, то можливо побудувати таку систему рекомендацій, яка буде давати максимальні результати якості. Метод експертної атрибутизації передбачає залучення знань про спортивну сферу для визначення відповідності товарів конкретним потребам користувачів. Це дозволяє створити семантично багату базу даних, яка є основою для контентної фільтрації. У споживчому маркетингу широко застосовується класифікація товарів широкого вжитку. Схильність до купівлі таких товарів найчастіше визначається купівельними звичками споживача. Використання правильної класифікації таких товарів дає змогу підприємству багато в чому визначитися щодо особливостей їхньої купівлі.

Доцільно використовувати метод зворотнього зв’язку від клієнта, коли покупець залишає відгук або оцінку на товар, це формує його ставлення до переглянутого або придбаного товару. Цей набір даних слугує інформацією для створення рекомендацій конретному користувачу.

2.1.2 Методи машинного навчання

Загалом машинне навчання використовую одну з двох методів. Перший метод контрольованого вивчення полягає в тому, що навчає модель на відомих вхідних і вихідних даних так,щоб це могло передбачити майбутні вихідні параметри, яке знаходить захованим шаблони або внутрішні структури у вхідних даних.

Мета машиного навчання з учителем полягає в тому, шоб створити модель, яка робить прогнози на основі доказу в присутності невпевненості (див. рис. 2.1)

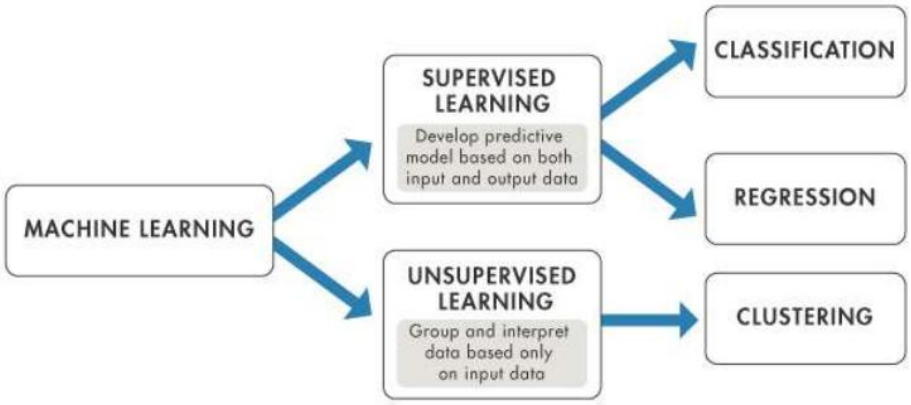


Рис. 2.1 Типи машинного навчання

Контрольоване вивчення використовує класифікацію та методи регресії [32].

Методи класифікації пророкують категоріальні відповіді, наприклад, чи є електронний лист справжнім або спам, або чи є пухлина злоякісною або доброякісною. Моделі класифікації класифікують вхідні дані в категорії. Типові програми включають медичну обробку зображень, зображення і розпізнавання мови і рейтинг кредитоспроможності.

Методи Регресії пророкують безперервні відповіді, наприклад, зміни в температурі або коливання споживання енергії. Типові програми включають прогнозування завантаження електрики і алгоритмічну торгівлю

2.1.3 Методи порівняльного аналізу та валідації

Для об'єктивної оцінки якості рекомендацій застосовується метод k-fold крос валідації. Наявні дані про взаємодії користувачів з товарами розділяються на k частин, де k-1 частина використовується для навчання моделі, а одна частина для тестування. Процедура повторюється k разів з різними комбінаціями навчальних та тестових даних, що дозволяє отримати усереднену оцінку ефективності та оцінити стабільність результатів. Альтернативно застосовується темпоральне розділення, коли старіші взаємодії використовуються для навчання, а нові для тестування, що відповідає реальному сценарію використання системи.

Для оцінки переваг розробленої системи проводиться порівняльний аналіз з базовими алгоритмами. До базових підходів відносяться рекомендації випадкових

товарів, найпопулярніших товарів, контентної фільтрації та колаборативної фільтрації окремо без гібридизації. Статистична значущість різниці у метриках оцінюється за допомогою t-тесту або інших статистичних критеріїв. Це дозволяє підтвердити, що покращення якості рекомендацій не є випадковим.

2.2 Теоретичні основи побудови рекомендаційних алгоритмів

2.2.1 Колаборативна фільтрація

Загалом підхід колаборативної фільтрації реалізовується як алгоритм аналізу вподобань інших користувачів системи [32]. Якщо двоє користувачів системи мають схожі оцінки для певного товара, то можна робити висновок, що цим двом клієнтам подобаються однакові групи товаром загалом.

Колабаративна фільтрація може бути застосована на користувачах системи, при цьому система шукає користувачів зі схожими інтересами та рекомендує їм товари, які комусь з них сподобалися. Або колабаративна фільтрація може бути заснована на об’єктах, при цьому алгоритм аналізує товари, які були високо оцінені схожими користувачами і показує і пробує продати їх іншим користувачам. Основані ці підходи на методі k найближчих сусідів. Шукаються користувачі, які мають схожі оцінки і на основі їх смаків рекомендує нові товари.

Ще може застосовуватися метод матричної факторизації. Цей підхід допомагає знайти приховані зв’язки між користувачами та товарами.

Основна перевага колабаративної фільтрації полягає у можливість рекомендації несподіваних варіантів, які навіть сам користувач не зміг би собі уявити, але при цьому вибір системи користувачу подобається. Недоліком цього методи є довгий час нормального його запуску, поки система не отримає достатню кількість інформації про користувачів та їх вподобання.

2.2.2 Контентна фільтрація

Контентний підхід основан на аналізі характеристик товарів в системі і вподобань покупців [33]. Якщо покупцю сподобався певний товар, йому можна

рекомендувати схожі товари. Для цього треба проаналізувати текстові описи, атрибути, категорію, рейтинг товару. При контент ній фільтрації можливо визначити, наскільки важливими для користувача є певні товари. Для визначення схожості між товарами використовують косинусну подібність. Цей метод дозволяє порівняти два товари та визначити, наскільки вони подібні своїм характеристикам.

Контентний підхід відмінно працює коли є достатньо інформації про товари. Цей підхід має ряд обмежень. Якщо користувач ще не переглядав жодного товару, система не може запропанувати релевантний варіант. Також при рекомендації цей підход не враховує думку інших юзерів.

2.2.3 Гібридні підходи

Гібрідний підхід поєднує в собі переваги контентного підходу та колаборативного підходу. При цьому компенсуються недостатки кожного з методів [34]. Комбінування методів можна отримати шляхом об’єднання результатів, коли система окремо генерує рекомендації за контент ним підходом і колаборативною фільтрацією, а потім їх поєднує. Ще буває каскадний підхід, коли один метод використовується як основний, а інший для уточнення результатів. Машинне навчання застосовується для автоматичного вибору найкращого методу рекомендацій для кожного конкретного користувача.

Таким чином, гібридні системи дозволяють отримати більш точні результати та дати більш різноманітніші рекоменаціх. Запустити можна гараздо швидше цей метод, навіть, якщо користувач новий.

2.2.4 Обґрунтування вибору алгоритму для розроблюваної системи

В рамках поточного проекту для створення системи рекомендацій при підбору спортивного спорядження використовується гібридний підхід, що поєднує контентну та колаборативну фільтрацію з адаптивним зважуванням компонентів. Такий вибір був зроблений тому, що система має особливу спиціфіку, в якій теба враховувувати не лише характеристики товарів, а й думки покупців спортивного спорядження. Також такий вибір ообгрунтовується собливостями взаємодії користувачів зі спортивним обладнанням.

Контентна фільтрація є в системі базовим компонентом. Цей метод є основним методом формування рекомендацій, особливо для нових користувачів. Поки система не бачить достатньої кількості дій від користувача, їй неможливо скласти рекомендації, тому просить продовжувати використовувати систему, щоб та вчилася. І щоб користувач відразу міг отримувати рекомендації, створено модуль профіля користувача, де він заповнює своє дані стосовно спорту. В профільну інформацію включаються види спорту, рівень підготовки, цілі тренувань, бюджетні преференції та участь у змаганнях.

Спортивне спорядження має чіткі технічні характеристики та призначення, що дозволяє ефективно застосовувати контентний підхід для зіставлення параметрів товарів з потребами користувача.

Реалізована система контентної фільтрації використовує зважену формулу оцінки релевантності з шістьма компонентами, де найбільшу вагу в 40% має відповідність виду спорту, що є критично важливим для спортивної сфери.

Другим за важливістю є рівень підготовки, що має вагу в 25%, оскільки використання спорядження невідповідного рівня може бути неефективним або навіть небезпечним.

Бюджетні преференції мають вагу в 15%, а цілі тренувань мають вагу 10%. Ці показники доповнюють профіль користувача, а участь у змаганнях з вагою в 5% та рейтинг товару з вагою в 5% додають контекстуальну інформацію.

Важливою перевагою контентного підходу в даному проєкті є можливість генерації пояснень рекомендацій. Система формує текстові пояснення, що товар

рекомендовано для бігу, для середнього рівня, середній сегмент. Також в системі виводиться інформація по цілям, що підвищує довіру користувачів та допомагає їм приймати обґрунтовані рішення про покупку.

Колаборативний компонент системи застосовується для виявлення несподіваних, але релевантних рекомендацій на основі досвіду схожих користувачів. Розроблена метрика схожості користувачів враховує три аспекти, серед яких схожість видів спорту, що має вагу в 40%, схожість параметрів профілю, що має вагу в 30% та схожість оцінок товарів, що має вагу в 30%.

Використання коефіцієнта Жаккара для порівняння множин видів спорту забезпечує ефективне виявлення користувачів зі схожими спортивними інтересами. Порівняння профілів за рівнем підготовки, бюджетними преференціями, цілями та участю у змаганнях дозволяє знайти користувачів з подібним спортивним стилем життя.

Кореляція Пірсона для оцінок товарів виявляє користувачів зі схожими смаками щодо конкретних характеристик спорядження. Колаборативна фільтрація особливо ефективна для користувачів з великою історією взаємодій, що налічує більше 15 оцінок, коли система може надійно визначити схожих користувачів та сформувати рекомендації на основі їхнього досвіду.

Ключовою особливістю розробленої системи є адаптивне зважування компонентів залежно від кількості даних про користувача. Для нових користувачів з кількістю оцінок менше 5 використовується розподіл ваг 80% контентна та 20% колаборативна фільтрація. Це дозволяє надавати релевантні рекомендації навіть без історії взаємодій, базуючись на детальному профілі користувача.

Для користувачів з середньою активністю, які ставили в системі товарам 5-14 оцінок застосовується баланс 60% на 40%, що забезпечує поступовий перехід до більшого використання колаборативної складової.

Для активних користувачів з 15 і більше оцінками розподіл змінюється на 40% на 60%, оскільки колаборативна фільтрація стає більш надійною при наявності достатньої кількості даних про уподобання користувача.

Така адаптивна стратегія вирішує проблему холодного старту, характерну для традиційних колаборативних систем, та забезпечує плавне покращення якості рекомендацій у міру накопичення даних про користувача.

Окрім точності, система забезпечує різноманітність рекомендацій через обмеження кількості товарів з однієї категорії, що становить максимум 2 товари та одного бренду, що становить максимум 2 бренди у списку рекомендацій. Це запобігає надмірній гомогенності пропозицій та дозволяє користувачу ознайомитися з ширшим спектром варіантів спорядження.

Для підвищення швидкодії системи реалізовано механізм попереднього розрахунку та збереження схожості товарів і користувачів у базі даних. Модуль забезпечує періодичний перерахунок матриць схожості, що дозволяє генерувати рекомендації в реальному часі без затримок на обчислення.

2.3 Обґрунтування технологічного стеку

2.3.1 Причини вибору Python

При розробці веб-додатків важливим аспектом є вибір відповідної мови програмування та фреймворку. Однією з найпопулярніших мов програмування для веб-розробки є Python, який має широкий набір бібліотек і фреймворків для швидкої та ефективної розробки [35].

Для створення веб-рішень на Python [36] широко використовують фремворки. Фреймворки Python — це набори модулів або пакетів, які допомагають розробникам створювати веб-додатки та інші програмні рішення. Існує чимало Python-фреймворків під різноманітні завдання.

Причиною вибору мови програмування Python для створення проекту системи рекомендацій спортивного спорядження став той факт, що Python дозволяє швидко створювати прототипи та розгортати вебдодатки завдяки фреймворкам, таким як Flask. Ця можливість особливо важлива для швидкого тестування нових ідей та функціональності. І другий факт вибору полягає в тому, шо динамічна

типізація в Python спрощує роботу з різними типами даних, що буде зручно для атрибутів спортивних товарів.

2.3.2 Огляд бібліотек

Однією з ключових переваг Python є його багата стандартна бібліотека, яка включає модулі для роботи з операційною системою, мережевими запитами, управлінням файлами та іншими завданнями. Це забезпечує широкі можливості без необхідності встановлення додаткових пакетів, спрощуючи розробку і забезпечуючи надійність [37].

Для того, щоб написати проект було обрано ряд бібліотек, які були підключені к проєкту та реалізують необхідний функціонал та підключення к базі даних.

Для того, щоб проект коректно працював з базою даних MySQL було додано бібліотеку SQLAlchemy. Ця бібліотека забезпечує об’єктно-реляційне відображення для взаємодії з MySQL базою даних.

Для спрощення конфыгурацыъ та використання SQLAlchemy було додано до проекту розширення Flask-SQLAlchemy. За допомогою цього розширення буде виконуватися автоматичне управління сеансами бази даних та інтеграцію з контекстом додатку.

Для виконання математичних операцій над векторами та масивами даних в проект додано зв’язок з бібліотекою NumPy. В розроблюваній системі рекомендацій ця бібліотека використовується для розрахунку статистичних показників, таких як середні значення оцінок, кореляція Пірсона та обчислення векторних відстаней.

Для реалізації машинного навчання використовується в проекті бібліотека scikit-learn. Ця бібліотека надає інструменти для аналізу даних та побудови моделей машинного навчання. В проекті ця бібліотека реалізовує розрахунок для оцінки точності прогнозування рейтингів користувачів.

Для фіксації часових міток в проекті використовується бібліотека Datetime. Ця бібліотека допомагає реалізувати фіксацію часових міток генерації рекомендацій, розрахунку схожості та аналізу метрик за певні періоди часу.

Для реалізації функціоналу авторизації та реєстрації використовується бібліотека Flask-Login. Вона допомагає керувати сеансами користувачів та надає декоратори для захисту маршрутів, що вимагають автентифікації та зберігає інформацію про поточного користувача та його сесію.

Для створення та валідації форм в проекті використовується бібліотека WTForms. Вона допомагає зробити автоматичну перевірку введених даних на відповідність правилам та запобігає атакам через ці форми, наприклад ін’єкціям.

Бібліотека Flask-Migrate виконує в проекті функцію управління міграціями бази даних. Вона контролює схеми бази даних. При зміні моделей даних зроблено так, щоб автоматично генерувалися файли міграції, які безпечно оновлюють структуру таблиць.

Бібліотека Python-dotenv дозволяє завантажувати змінні середовища. Це забезпечує зручність в управлінні конфігураційними параметрами проекту.

2.3.3 Обґрунтування вибору Flask як серверної платформи

Flask – це легкий мікро-фреймворк для створення веб-додатків, який забезпечує базовий набір функцій для розробки [38]. Він підтримує розширення для валідації форм, об'єктно-реляційного відображення, автентифікації та завантаження ресурсів, а також надає низку інших корисних інструментів.

Основні переваги Flask:

- вбудована підтримка модульного тестування;
- можливість надсилання RESTful-запитів;
- використання шаблонізатора Jinja2;
- підтримка безпечних файлів cookie для збереження клієнтських сеансів;
- детальна та зрозуміла документація;
- можливість експериментувати з модулями чи архітектурою;

– підходить для невеликих проектів;
– легко масштабується;
– сумісність із Google App Engine.

Фреймворк Flask використовує механізм шаблонів для того, щоб завантажувати на клієнтській частині інформацію динамічно, створюючи сторінки в браузері.

Для встановлення фреймворка в проєкт потрібно виконати команду інсталяції в терміналі (див. рис. 2.2).

```
(.venv) PS D:\sports_recommendation_system> pip install flask
```

Рис. 2.2

Встановлення фреймоврка Flask

Після виконання цієї команди проект буде включати всі необхідні компоненти для роботи з фреймворку Flask.

2.3.4 Обґрунтування вибору MySQL

Додаток націлено на взаємодію із базою даних, отже для їх створення було обрано систему управління реляційними базами даних MySQL. Саме ця СУБД була використана через її здатність швидкої обробки інформації, легкість використання. За допомогою phpMyAdmin було здійснено управління та конфігурування бази даних.

MySQL забезпечує високу швидкість обробки запитів, що критично важливо для системи рекомендацій, де необхідно виконувати складні вибірки даних для розрахунку схожості користувачів і товарів. Реляційна модель даних ідеально підходить для структурованої інформації про профілі користувачів, характеристики товарів та їхні взаємозв'язки. MySQL підтримує транзакції та

обмеження цілісності даних, що гарантує консистентність інформації при одночасних операціях багатьох користувачів.

При виборі СУБД розглядалися альтернативи, такі як PostgreSQL та SQLite. PostgreSQL пропонує більш розширені можливості для складних запитів та підтримує просунуті типи даних, однак для поточної версії проєкту функціоналу MySQL цілком достатньо та я маю досвід роботи саме із цією СКБД.

2.3.5 Додаткові інструменти розробки та тестування

Для свого стеку технологій я використовую Wampserver64 (див. рис. 2.3) як сервер, phpMyAdmin як інтерфейс для взаємодії з базою даних, СУБД MySQL.

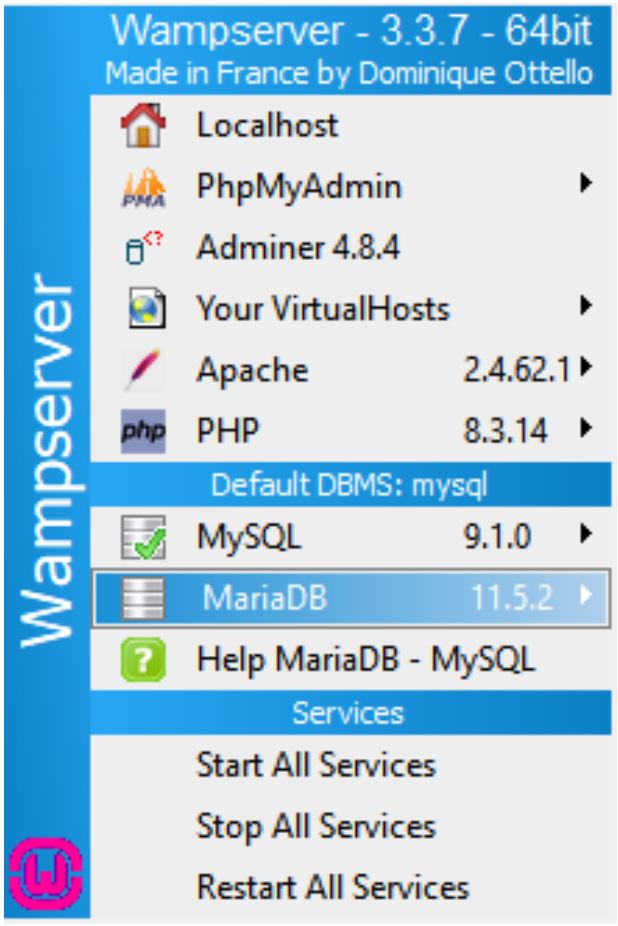


Рис. 2.3 Запуск WampServer

WampServer дозволяє швидко розгортати локальне середовище розробки без необхідності окремого встановлення та налаштування кожного компонента.

Панель управління WampServer надає можливість швидкого запуску, зупинки та перезапуску сервісів, перегляду логів помилок та конфігурування параметрів серверів. Зелений індикатор у системному треї свідчить про успішний запуск усіх сервісів та готовність середовища до роботи.

Для написання коду та інсталяції бібліотек та розширень був використаний редактор коду PyCHarm (див. рис. 2.4).

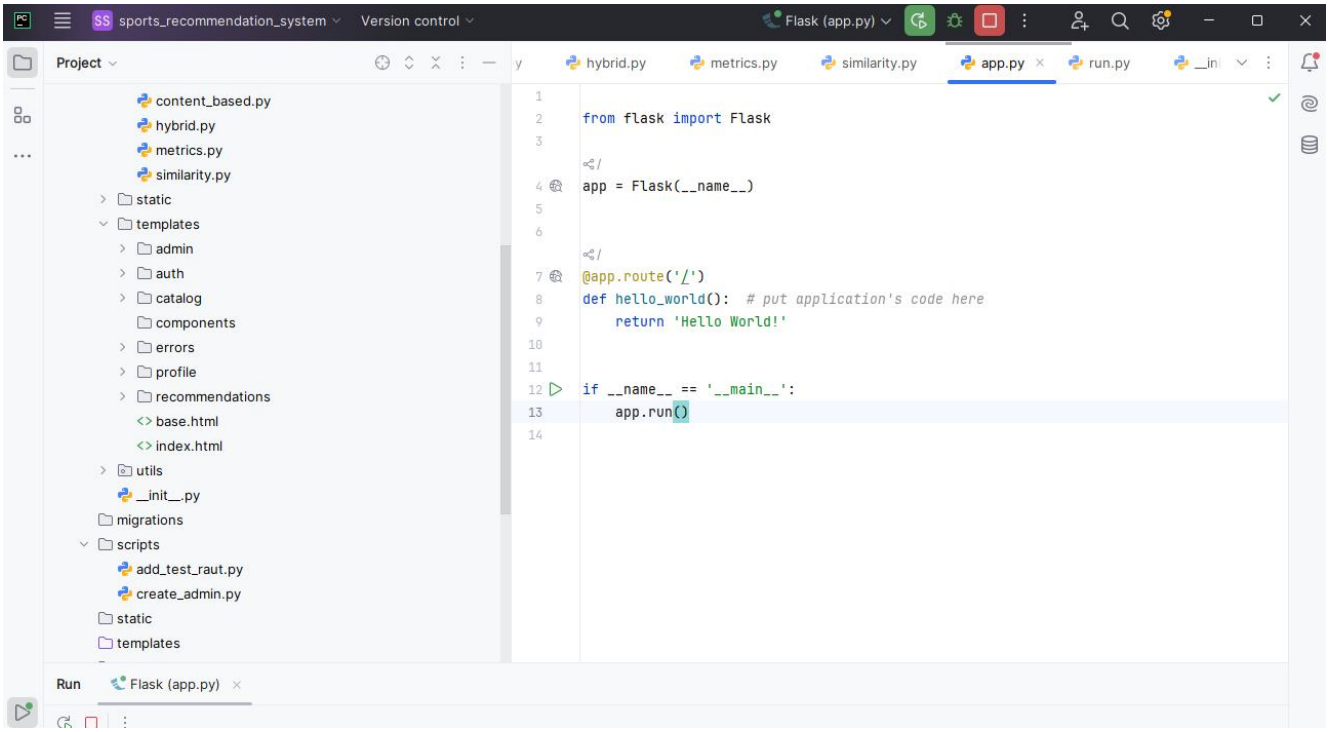


Рис. 2.4 Середовище розробки PyCharm

PyCharm є професійним IDE від компанії JetBrains, спеціально розробленим для Python-розробки, і надає широкий спектр інструментів для підвищення продуктивності розробника. PyCharm аналізує структуру проєкту та надає контекстно-залежні підказки при написанні коду. Це включає автодоповнення назв змінних, методів, класів та імпортів, що значно прискорює розробку та зменшує кількість помилок друку.

3 ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Формування вимог до системи

Формування вимог до системи представляє собою технічне завдання для розробки системи. В даном пункті формуються функціональні і не функціональні вимоги на основі потреб користувачів, специфіки предметної галузі та технічних можливостей вибраного технологічного стеку.

3.1.1 Функціональні вимоги

Функціональні вимоги до системи описують, що саме зможе робити системи, які її конкретні можливості та як вона вміє взаємодіяти з користувачем. В системі мають бути реалізовані наступні модулі:

- Модуль управління користувачами та автентифікація;
- Модуль спортивного профілювання користувача;
- Модуль каталогу товару;
- Модуль рекомендацій;
- Модуль історії взаємодії;
- Модуль адміністрування;
- Модуль аналітики та метрик.

Система має розділяти ролі користувача на адміністратора та покупця. Авторизація в системі має відбуватися через форму авторизації. Авторизований адміністратор має мати доступ до керування товарами, а саме мати можливість створювати нові товари, редагувати існуючі товари, видаляти товари та редагувати список доступних категорій, до яких належать товари. Адміністратор має мати можливість виконувати верифікацію відгуків, які поступають в систему, які він зможе видаляти, якщо відгук суперечить правилам сайту. Відгуки мають проходити модерацію і відображати один зі статусу, такі як на модерації, схвалені.

Також адміністратору має бути доступен модуль аналітики, де адміністратор може побачити топ-10 товарів за переглядами, статистику ефективності кліків на основі профілю користувача, на основі схожості, результат рекомендації гібридної моделі.

Звичайний користувач має мати можливість переглянути каталог спортивного інвентарю. Для того, щоб поставити оцінку товару, чи написати відгук на товар, система має направити користувача на сторінку реєстрації, щоб той створив свій профіль на платформі. Під час створення профілю користувач має вказати свої анкетні дані, а саме фізичні дані, вподобання до виду спорту і спортивні досягнення. Після цього користувач має мати можливість авторизуватися в системі, переглянути каталог товарів, додати відгук на товар, оцінии його, додати товар в обране, переглянути сторінку рекомендацій.

Створення та формування спортивного профілю користувача має включати наступні теми:

- Вибір одного або декількох видів спорту з позначенням пріоритету
- Рівень підготовки
- Цілі тренувань з можливістю множинного вибору
- Частота тренувань
- Участь у змаганнях з можливістю вказати типи змагань
- Фізичні параметри
- Бюджетні преференції

Користувач повинен мати можливість редагувати свій профіль у будь-який час, і зміни повинні автоматично враховуватися при наступній генерації рекомендацій.

Для кожного товару повинна відображатися основна інформація: назва, фото, ціна, бренд, короткий опис, рейтинг та кількість відгуків. Перегляд каталогу повинен містити функцію фільтрації для зручності використання користувачем. Фільтрація повинна бути реалізована багатокритеріальна з наступними параметрами:

- Вид спорту

- Тип товару
- Ціновий діапазон
- Бренд
- Рівень підготовки користувача
- Призначення
- Рейтинг

Система повинна підтримувати текстовий пошук за назвою товару та характеристиками з можливістю комбінування з фільтрами.

Користувач повинен мати можливість сортувати товари за різними критеріями: за популярністю, за рейтингом, за ціною, за новизною надходження до каталогу.При переході на сторінку конкретного товару користувач повинен бачити повну інформацію: детальний опис, технічні характеристики, матеріали, розмірну сітку.

Система повинна генерувати персоналізовані рекомендації на основі параметрів спортивного профілю користувача з використанням алгоритму контентної фільтрації. Рекомендації повинні враховувати види спорту, рівень підготовки, цілі тренувань, бюджетні преференції та участь у змаганнях користувача. Система повинна надавати блок рекомендацій на основі схожих користувачів, використовуючи алгоритм колаборативної фільтрації.

3.1.2 Нефункціональні вимоги

Нефункціональні вимоги описують якісні властивості системи, які в комплексі створюють зручність використання та ефективність взаємодії із системою. Все, що стосується продуктивності, безпеки, зручності використання, все це сформовано у не функціональних вимогах:

- Генерація списку рекомендацій повинна виконуватися не більше 2 секунд для гібридного алгоритму
- Відображення сторінки каталогу з фільтрами повинно займати не більше 1 секунди

- Пошук товарів повинен повертати результати не більше ніж за 1.5 секунди
- Схожість між товарами та користувачами повинна розраховуватися попередньо та зберігатися в базі даних
- Попередній розрахунок схожості повинен виконуватися автоматично щоночі для нових товарів та користувачів
- Персональні рекомендації користувача повинні кешуватися на певний період для зменшення навантаження
- Кеш рекомендацій повинен автоматично інвалідуватися при зміні профілю користувача або додаванні нових оцінок
- Система повинна підтримувати одночасну роботу щонайменше 100 користувачів без значного погіршення продуктивності
- Алгоритми рекомендацій повинні ефективно працювати з каталогом до 10,000 товарів та базою до 5,000 користувачів
- Паролі користувачів повинні зберігатися виключно у хешованому вигляді з використанням bcrypt
- Всі запити до бази даних повинні виконуватися через ORM з параметризованими запитами
- Всі форми повинні містити CSRF-токени для захисту від міжсайтової підробки запитів
- Весь користувацький контент повинен проходити санітизацію перед відображенням
- У продакшн-середовищі система повинна працювати виключно через захищене HTTPS-з'єднання
- Сеанси користувачів повинні мати обмежений час життя з автоматичним завершенням при неактивності
- Система повинна підтримувати примусове завершення всіх активних сеансів при зміні пароля
- Система повинна надавати користувачам можливість переглядати, експортувати та видаляти свої дані

- Користувач повинен мати можливість дістатися до будь-якої сторінки максимум за 3 кліки
- При виникненні помилок система повинна відображати зрозумілі повідомлення з рекомендаціями щодо усунення
- Система повинна підтримувати українську та англійську мови інтерфейсу
- Навігація повинна бути інтуїтивно зрозумілою для користувачів будь-якого рівня технічної підготовки
- Система повинна забезпечувати доступність не менше 99% часу, виключаючи заплановані технічні роботи
- Система повинна коректно обробляти виняткові ситуації без аварійного завершення роботи
- Критичні помилки повинні логуватися для подальшого аналізу
- Система повинна мати можливість відновлення після збою серверу протягом максимум 1 години

Дотримання всіх вищезазначених не функціональних вимог до системи дасть можливість системі бути якісною та привабливою для користувачів.

3.1.3 Вимоги до якості даних

Якість даних безпосередньо впливає на точність рекомендацій та загальну ефективність системи. Вимоги до якості даних охоплюють структуру, повноту, консистентність та актуальність інформації.

- При створенні спортивного профілю користувач повинен обов'язково заповнити хоча б один вид спорту, рівень підготовки, бюджетні преференції
- Фізичні параметри повинні перевірятися на реалістичність
- Кожен товар повинен мати обов'язкові атрибути назва, опис, основне фото, ціна, вид спорту, категорія, рівень підготовки, ціновий сегмент
- Товари повинні класифікуватися за стандартизованою ієрархією
- Ціна товару повинна бути позитивним числом з точністю до двох десяткових знаків

- Для товарів основних категорій повинен бути вказаний зв'язок з цілями тренувань
- Недоступні товари повинні деактивуватися замість видалення для збереження історії

Що стосується оцінок та відгуків, то тут має бути враховано, що кожен користувач може залишити одну оцінку для одного товару. Ця оцінка має бути цілим числом від 1 до 5. Всі оцінки мають бути зафіксовані за часом їх додавання.

3.2 Проєктування архітектури системи

3.2.1 Загальна архітектурна схема

Система рекомендації товарів спортивного спорядження для користувачів сайту побудована як класична система з архітектурою тришарового веб-додатку. Таким чином забезпечується чітка декомпозиція система на слої, розділяється відповідальність між компонентами.

В системі реалізовано рівень представлення, рівень бізнес логіки, рівень доступу до даних та рівень збереження до даних.

Клієнтська частина, що відповідає рівню представлення, написана як шаблони веб-додатку з використанням html, css, bootstrap та JavaScript.

Серверна частина, що відповідає рівню бізнес логіки реалізована на мові Python з використанням фремворка Flask. Тут написані всі контролери, що обробляють запити з клієнтської частини. Всі модулі є незалежними за рахунок концепції інкапсуляції.

Рівень доступу до даних представляє собою взаємодію з базою даних, яка реалізована через ORM SQLAlchemy. Таким чином забезпечується об’єктно-реляційне відображення. Всі моделі визначені в системі та відповідають основним сутностям системи. SQLAlchemy автоматично генерує SQL запити на основні операції з моделями.

Рівень збереження даних організована у вигляді зберігання даних в реляційній базі даних MySQL. В базі даних створені всі таблиці, що відповідають

сутностям системи. Загальна архітектура системи реалізована у вигляді ієрархії файлів та папок в проекті (див. рис. 3.1).

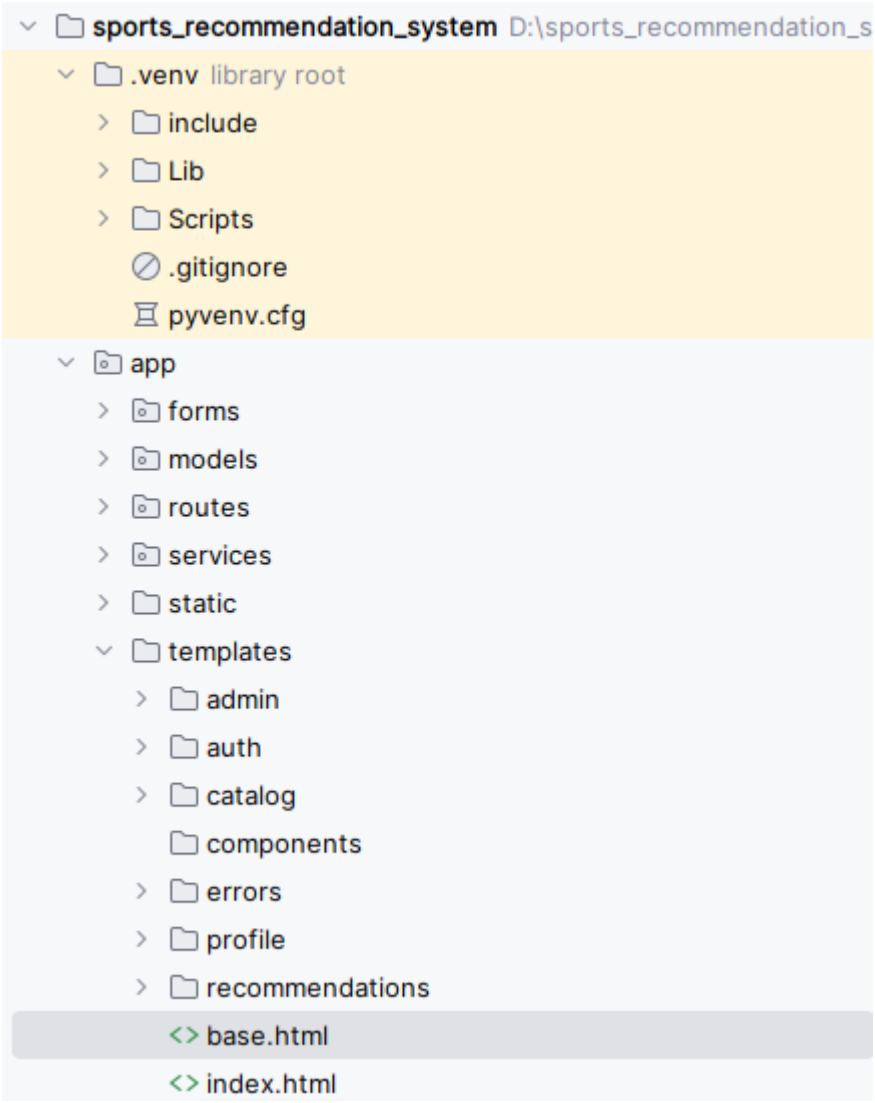


Рис. 3.1 Ієрархія файлів та папок проектів

Код всього проекту розділений на логічні модулі та пакети. Система включає компоненти для автоматизації та оптимізації роботи. Переодично можна запускати модуль для навчання системи. Тришарова архітектура системи включає в себе механізми безпеки на кожному рівні. Також вона дозволяє легко масштабувати систему, в будь-який момент часу додавши новий модуль.

3.2.2 Діаграми UML

Діаграма варіантів використання ідентифікує основних акторів системи, таких як адміністратор, неавторизований користувач, авторизований користувач та система рекомендаці (див. рис. 3.2).

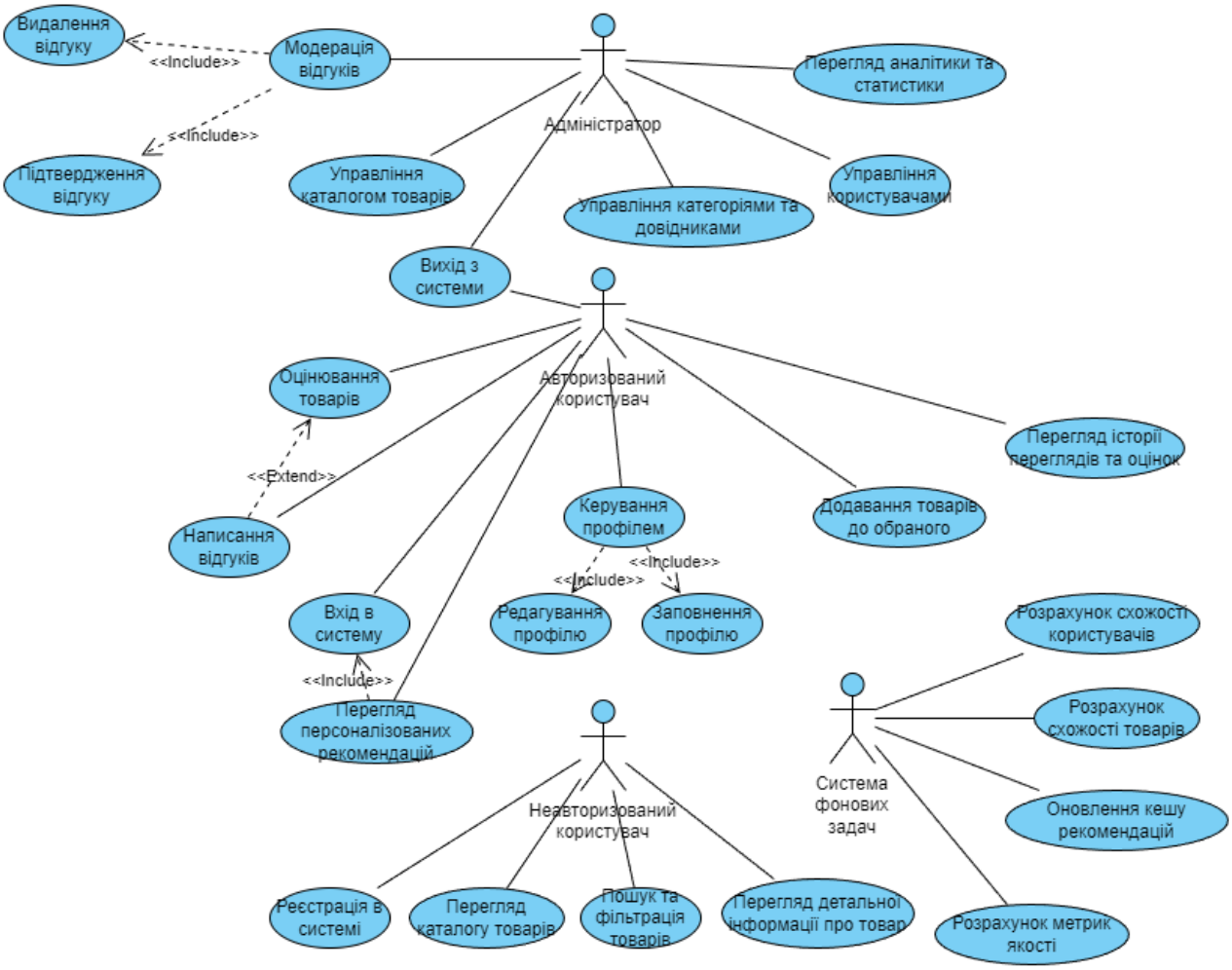


Рис. 3.2 Use-case діаграма

Таким чином відбувається взаємодія користувачів із модулями системи. Для того, щоб більш детально зрозуміти, як працює система, створена діаграма класів (див. рис. 3.3). Класи описують програмно сутності системи, які реалізовані в системі. Класи мають свої поля та методи.

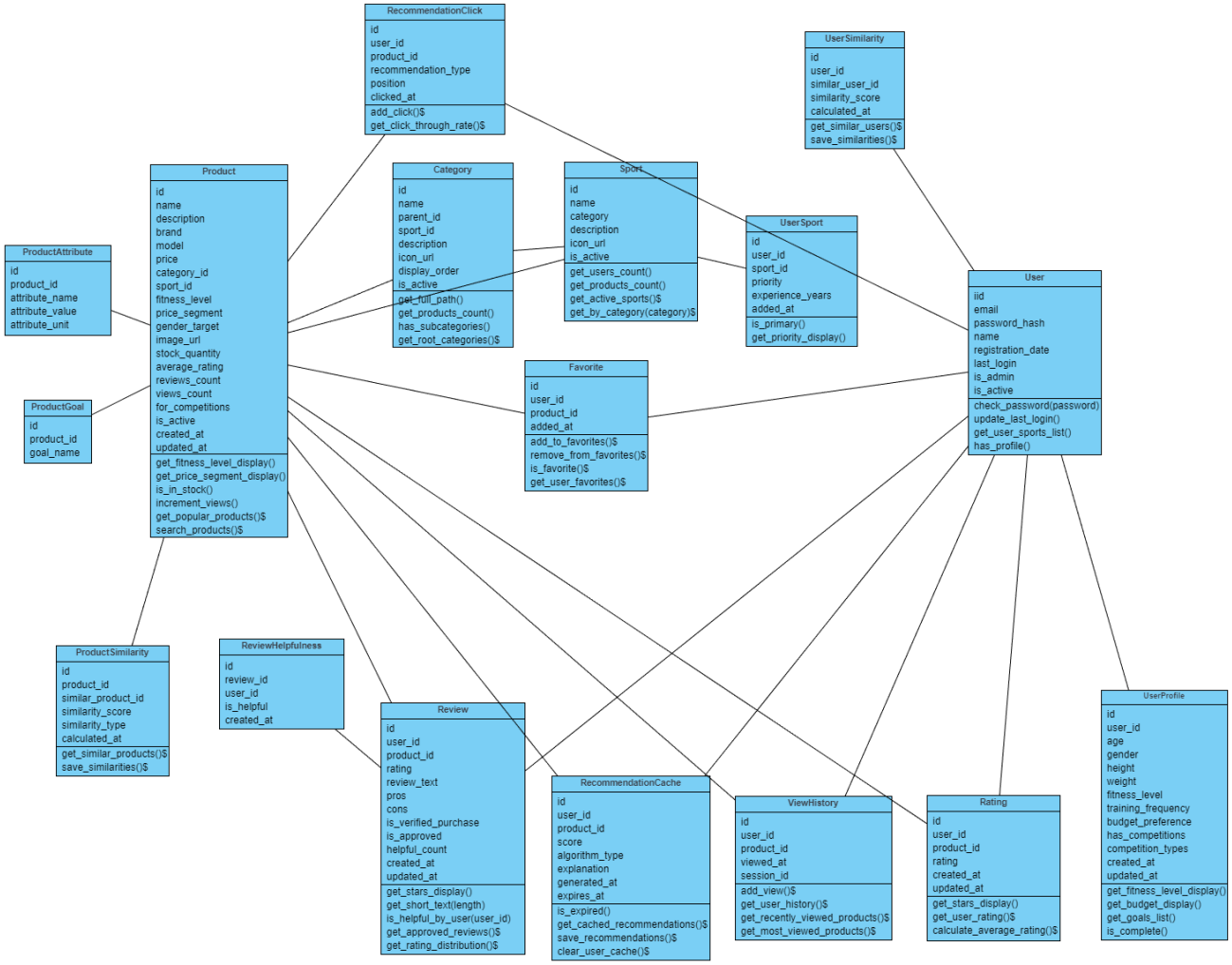


Рис. 3.4 Діаграма класів

Архітектура системи спроектована таким чином, що в будь-який момент можна додати новий модуль, навіть окремо його протестувати, при цьому не зачепити сусідні модулі чи систему цілком. Багатошарова архітектура дозволяє легко масштабувати проект та оновлювати в майбутньому. В архітектурі присутній рівень представлення, рівень бізнес логіки та рівень даних. Обмін інформацією реалізує рівень даних. При цьому система генерує запити до бази даних та отримає дані у відповідь (див. рис. 3.5).

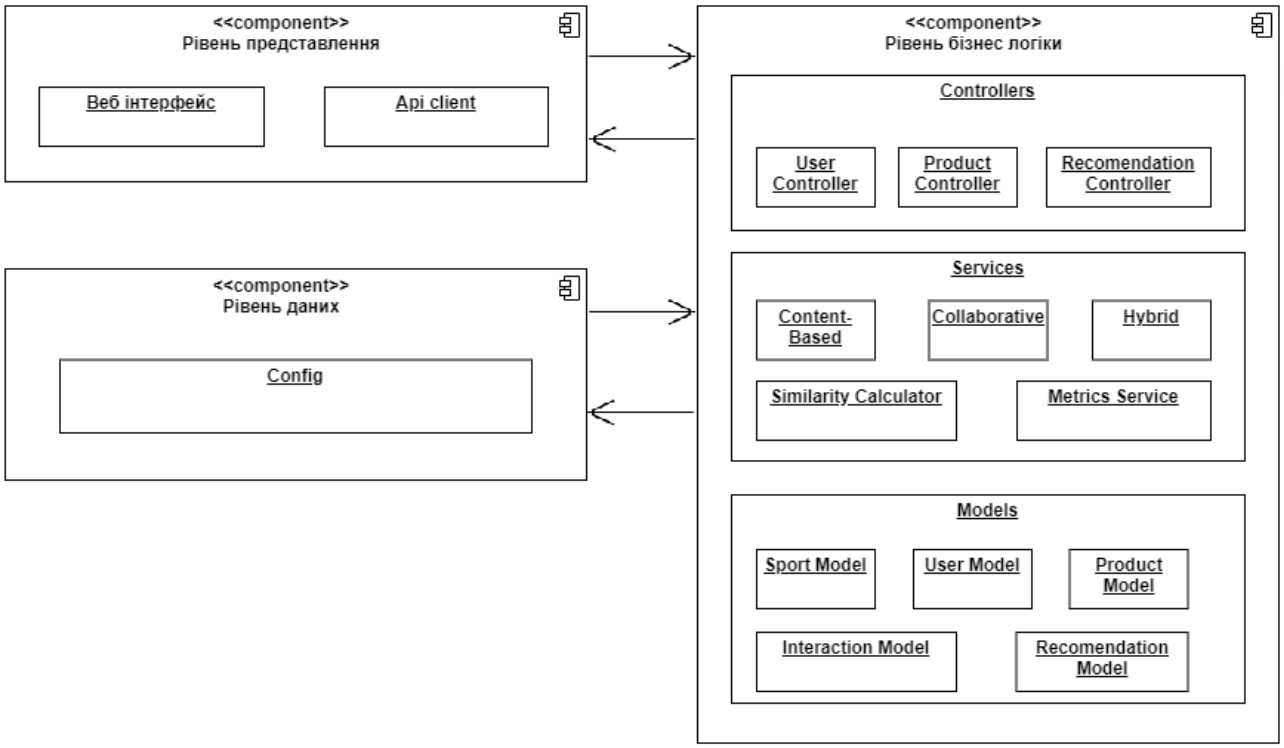


Рис. 3.5 Діаграма компонентів системи

Така архітектура має високу гнучкість та легку оновлюваність. База даних, яка використовується працює на СКБД MySQL. В наступному підрозділі буде спроектована концептуальна модель бази даних, логічна модель бази даних та фізична модель бази даних.

3.2.3 Визначення потоків даних

Контекстна діаграма 0 рівня (див. рис. 3.6) представляє систему рекомендацій спортивних товарів як єдиний процес, що взаємодіє з зовнішнім середовищем. На цьому рівні показано двох основних акторів. Користувач шукає персоналізовані рекомендації спортивних товарів. Адміністратор управляє каталогом та моніторить ефективність системи. Користувач надає свої преференції, профільні дані та взаємодіє з товарами, отримуючи натомість персоналізовані рекомендації та інформацію про товари. Адміністратор забезпечує систему даними про товари та отримує аналітичні звіти про роботу рекомендаційних алгоритмів.

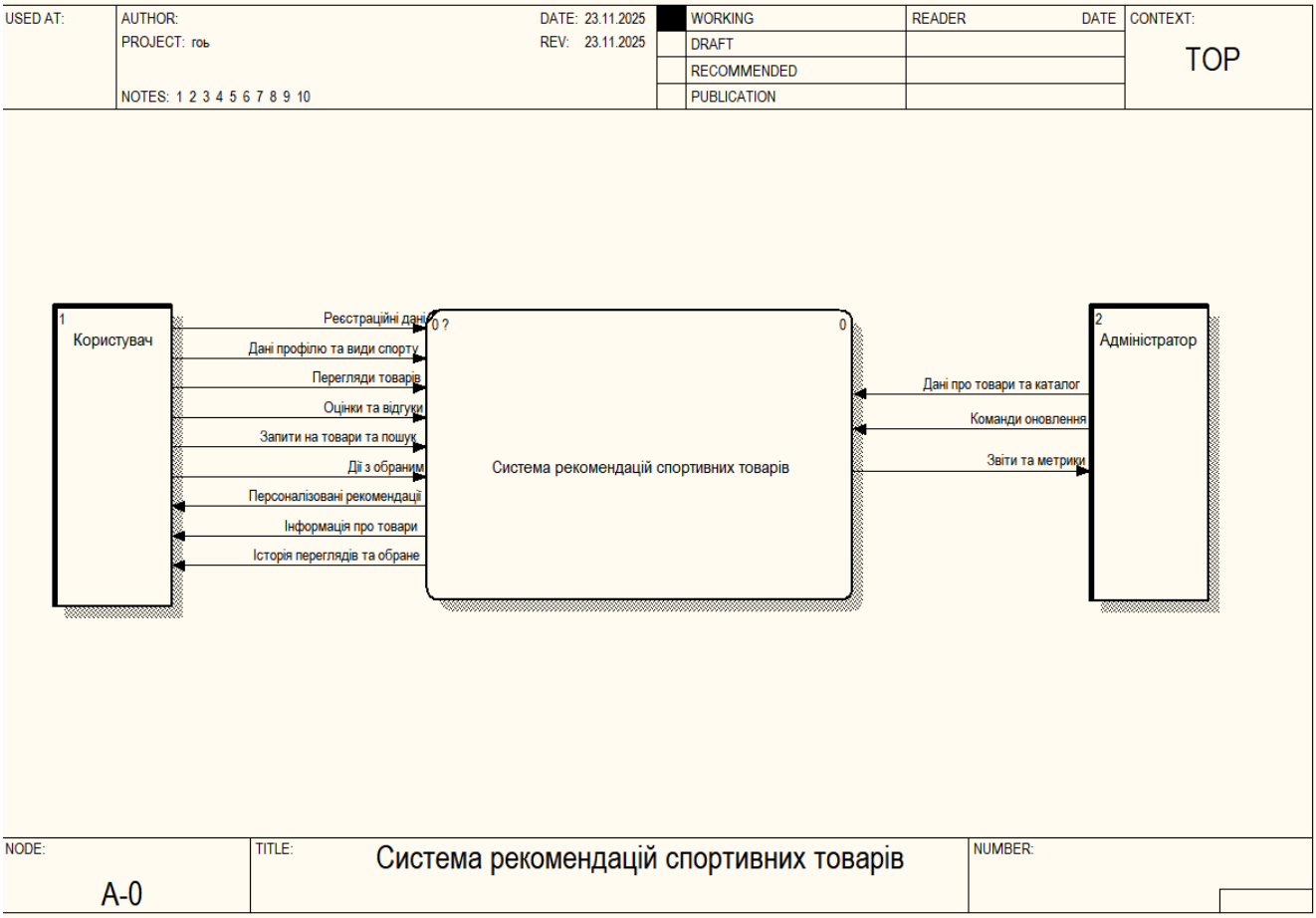


Рис 3.6 – Діаграма потоків даних 0 рівня

Діаграма рівня 1 декомпонує систему на шість ключових процесів, які реалізують функціональність рекомендаційної системи. Процеси включають управління користувачами та їх профілями, управління каталогом товарів, збір та обробку взаємодій користувачів з товарами, генерацію персоналізованих рекомендацій, розрахунок схожостей для оптимізації роботи та аналітику ефективності. На цьому рівні також показано 14 сховищ даних, які зберігають інформацію про користувачів, товари, взаємодії та розраховані метрики. Потоки даних між процесами демонструють, як інформація циркулює через систему: від збору даних про користувача через генерацію рекомендацій на основі Content-Based та Collaborative Filtering підходів до оцінки якості рекомендацій та покращення алгоритмів.

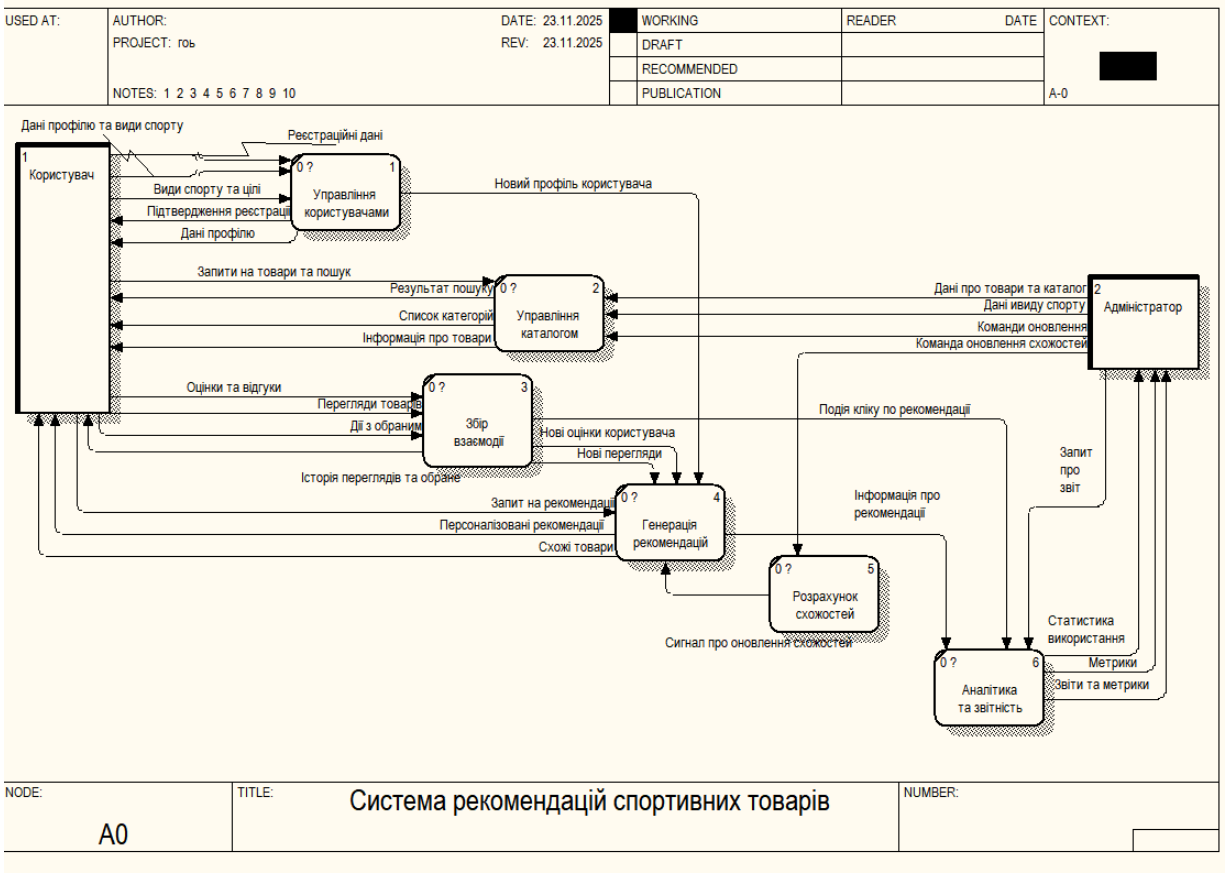


Рис. 3.7 Діаграма потоків даних першого рівня

Декомпозиція діаграми потоків даних була виконана на шість під процесів, що дозволило показати основні потоки даних на рівні функціонування системи та відобразити взаємні потоки даних між під процесами.

3.3 Проєктування бази даних

3.3.1 Концептуальна модель

Для розуміння концептуальної моделі організації даних в системі доцільно виділити сутності системи та кластерезувати їх у групи (див. рис. 3.8).

Рис. 3.8 Сутності системи, виявлені при концептуальному моделюванні

Основну концептуальну модель становлять чотири ключові групи сутностей. Користувачі та їх профілі це головний актив системи, бо без клієнтів не буде працювати і бізнес взагалі. Далі потрібна сутність каталогу товарів та видів спорту, щоб користувачі могли якось обирати товари та сортувати їх за видами спорту. Для навчання системи та генерації рекомендацій від системи до користувача потрібно реалізувати групу сутностей системи оцінок і відгуків, щоб система могла на чомусь навчатися. Та потрібна група сутностей механізмів рекомендацій та аналітики.

Центральною, як вже було сказано, сутністю є користувач, він же клієнт, він же відвідувач сайту. Кожен користувач має базовий заповнений профіль та загальну інформацію для авторизації. В анкеті користувач вказує, якими видами спорту він займається чи захоплюється. Тут можна вказати один або декілька варіантів. Тобто вид спорту це та сутність, яка зберігає варіанти видів спорту для вибора користувачем. Також користувач може виставити пріоритетність видам

спорту та вказати, скільки часу він займається кожним з них. Це вказується в системі через асоціативну сутність спорт користувача.

Далі користувач має шось купляти в системі. Товарна частина моделі представлена сутностями товар та категорія. Категорія організована як ієрархічна структура та пов’язана з видами спорту. Для збереження додаткових характеристик товара реалізована сутність атрибуту товару. Зв’язок товарів з цілями тренувань, які вказує користувач у своєму профілі реалізован у сутності цілі товару, що далі при роботі системи дозволить рекомендувати спорядження відповідно до фітнес цілей користувачів.

Механізм збору реакцій на товар від користувача реалізовано через сутності оцінок та відгуів. Для підвищення якості відгуків ще створено сутність корисність відгуку, щоб дивитися, як відгук впливає на реальну відповідність думки. Історія взаємодії користувача з товарами відображається у сутності історії переглядів та сутності обране. Це буде важливе джерело даних для алгоритмів рекомендацій. Алгоритми будуть працювати і рекомендувати шось користувачу. Для того, щоб оптимізувати роботу, в сутності кеш рекомендацій будуть описані попередньо розраховані рекомендації з оцінками релевантності та поясненнями. Також для алгоритму потрібні допоміжні сутності схожість товарів та схожість користувачів. Ці сутності служать для того, щоб прискорювати роботу алгоритму, зберігаючи попередні обчислення коефіцієнтів подібності.

Сутность кліки на рекомендації фіксує взаємодію користувача з рекомендаціями, щоб потім переглянути адміністратору статистику ефективності. Аналітична складова моделі представлена сутностями системні логи та метрики рекомендацій.

3.3.2 Логічна модель

Таблиця користувачів в базі даних буде зберігати основну інформацію про зареєстрованого юзера сайту і буде містити дані для авторизації та дані заповненої анкети, де вказані його вподобання (див. табл. 3.1).

Таблиця 3.1

Атрибути таблиці користувача на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор користувача	INT (PK, AUTO_INCREMENT)
email	Електронна пошта для входу	VARCHAR(255), UNIQUE
password_hash	Хеш пароля	VARCHAR(255)
name	Ім'я користувача	VARCHAR(100)
registration_date	Дата реєстрації	DATETIME
last_login	Дата останнього входу	DATETIME
is_admin	Чи є адміністратором	BOOLEAN
is_active	Чи активний акаунт	DATETIME

Сутність профіля користувача утворює відповідну таблицю, яка містить детальну інформацію про юзера для побудови рекомендацій. Тут зберігається все те, шо користувач заповнить при створені анкети (див. табл. 3.2).

Таблиця 3.2

Атрибути таблиці профіля користувача на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор профілю	INT (PK, AUTO_INCREMENT)
user_id	Зовнішній ключ на користувача	INT (FK users.id), UNIQUE
age	Вік користувача	INT
gender	Стать	ENUM('male', 'female', 'other')
height	Зріст у сантиметрах	DECIMAL(5,2)
weight	Вага у кілограмах	DECIMAL(5,2)
fitness_level	Рівень фізичної підготовки	ENUM
training_frequency	Кількість тренувань на тиждень	INT
budget_preference	Бюджетні переваги	ENUM
has_competitions	Чи бере участь у змаганнях	BOOLEAN
competition_types	Типи змагань	JSON
goals	Цілі тренувань	JSON
created_at	Дата створення профілю	DATETIME
updated_at	Дата останнього оновлення	DATETIME

Сутність видів спорту представляє собою довідник спортивної діяльності, які можно вибрати у системі. І потрібна ця таблиця для класифікації товарів за інтересами користувачів (див. табл. 3.3).

Таблиця 3.3

Атрибути таблиці види спорту користувача на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор виду спорту	INT (PK, AUTO_INCREMENT)
name	Назва виду спорту	VARCHAR(100), UNIQUE
category	Категорія виду спорту	VARCHAR(50)
description	Опис виду спорту	TEXT
icon_url	Посилання на іконку	VARCHAR(255)
is_active	Чи активний вид спорту	BOOLEAN

Сутність спорт користувача буде проміжною таблицею, яка утворюється через зв’язок багато до багатьох і зв’язує користувачів із видами спорту, якими вони займаються. Кожен користувач може вибрати один або декілька видів спорту (див. табл. 3.4).

Таблиця 3.4

Атрибути таблиці спорт користувача на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK, AUTO_INCREMENT)
user_id	Зовнішній ключ на користувача	INT (FK users.id)
sport_id	Зовнішній ключ на вид спорту	INT (FK sports.id)
priority	Пріоритет виду спорту	ENUM
experience_years	Досвід у роках	INT
added_at	Дата додавання	DATETIME

Таблиця категорії зберігає в собі інформацію про категорію товарі і допускає створення вкладених категорій для організації ієрархічної структури категорій товарів у системі (див. табл. 3.5).

Таблиця 3.5

Атрибути таблиці категорії на логічному рівні

Назва атрибуту	Опис	Тип даних

id	Унікальний ідентифікатор категорії	INT (PK)
name	Назва категорії	VARCHAR(100)
parent_id	Зовнішній ключ на батьківську категорію	INT (FK categories.id)
sport_id	Зовнішній ключ на вид спорту	INT (FK sports.id)
description	Опис категорії	TEXT
icon_url	Посилання на іконку	VARCHAR(255)
display_order	Порядок відображення	INT
is_active	Чи активна категорія	BOOLEAN

Сутність товари реалізована таблицею товари, яка є основою для каталогу спортивного спорядження. Тут зберігається вся інформація про товари, їх характеристики (див. табл. 3.6).

Таблиця 3.6

Атрибути таблиці товари на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор товару	INT (PK)
name	Назва товару	VARCHAR(255)
description	Опис товару	TEXT
brand	Бренд виробника	VARCHAR(100)
model	Модель товару	VARCHAR(100)
price	Ціна товару	DECIMAL(10,2)
category_id	Зовнішній ключ на категорію	INT (FK categories.id)
sport_id	Зовнішній ключ на вид спорту	INT (FK sports.id)
fitness_level	Рівень підготовки	ENUM
price_segment	Ціновий сегмент	ENUM
gender_target	Цільова стать	ENUM
image_url	Посилання на зображення	VARCHAR(255)
stock_quantity	Кількість на складі	INT
average_rating	Середній рейтинг	DECIMAL(3,2)
reviews_count	Кількість відгуків	INT
views_count	Кількість переглядів	INT
for_competitions	Чи для змагань	BOOLEAN
is_active	Чи активний товар	BOOLEAN
created_at	Дата створення	DATETIME
updated_at	Дата оновлення	DATETIME

В таблиці атрибутів товарів зберігаються додаткові характеристики товарів. Через цю таблицю можна додавати специфічні параметри для різних типів спорядження (див. табл. 3.7).

Таблиця 3.7

Атрибути таблиці категорії на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор атрибуту	INT (PK,)
product_id	Зовнішній ключ на товар	INT (FK products.id)
attribute_name	Назва характеристики	VARCHAR(100)
attribute_value	Значення характеристики	VARCHAR(255)
attribute_unit	Одиниця виміру	VARCHAR(50)

Таблиця цілі товарів утворюється через зв’язок багато до багатьох і є проміжною таблицею між товарами та тренувальними цілями. Один товар може підходити для використанні з декількома цілями (див. табл. 3.8).

Таблиця 3.8

Атрибути таблиці цілі товара на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
product_id	Зовнішній ключ на товар	INT (FK products.id)
goal_name	Назва тренувальної мети	VARCHAR(100)

Таблиця оцінок товарів показує рейтинг товару від користувача за шкалою від 1 до 5. Кожен користувач може оцінити товар лише один раз (див. табл. 3.9).

Таблиця 3.9

Атрибути таблиці оцінки на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор оцінки	INT (PK)
user_id	Зовнішній ключ на користувача	INT (FK users.id)
product_id	Зовнішній ключ на товар	INT (FK products.id)
rating	Оцінка від 1 до 5	TINYINT
created_at	Дата створення оцінки	DATETIME

Для сутності відгуків на товари створена відповідна таблиця на логічному рівні, яка містить інформацію про відгук з описом переваг та недоліків (див. табл. 3.10).

Таблиця 3.10

Атрибути таблиці відгуків на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор відгуку	INT (PK)
user_id	Зовнішній ключ на користувача	INT (FK users.id)
product_id	Зовнішній ключ на товар	INT (FK products.id)
rating	Оцінка від 1 до 5	TINYINT
review_text	Текст відгуку	TEXT
pros	Переваги товару	TEXT
cons	Недоліки товару	TEXT
is_verified_purchase	Чи підтверджена покупка	BOOLEAN
is_approved	Чи схвалений модератором	BOOLEAN
helpful_count	Кількість позитивних оцінок	INT
created_at	Дата створення	DATETIME
updated_at	Дата оновлення	DATETIME

Для того, щоб реалізувати оцінювання користувачами інших відгуків від інших користувачів організована таблиця корисність відгуків. Ця таблиця допомагає виявити найбільш інформативні відгуки (див. табл. 3.11).

Таблиця 3.11

Атрибути таблиці корисність відгуків на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
review_id	Зовнішній ключ на відгук	INT (FK reviews.id)
user_id	Зовнішній ключ на користувача	INT (FK users.id)
is_helpful	Чи корисний відгук	BOOLEAN
created_at	Дата оцінки	DATETIME

Для того, щоб система могла брати дані для алгоритму рекомендації про історії перегляду та замовлень користувача, створена таблиця історії переглядів, яка зберігає всю історію переглядів користувача для подальшого аналізу поведінки та побудови рекомендацій (див. табл. 3.12).

Таблиця 3.12

Атрибути таблиці історія переглядів на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
user_id	Зовнішній ключ на користувача	INT (FK users.id)
product_id	Зовнішній ключ на товар	INT (FK products.id)
viewed_at	Дата та час перегляду	DATETIME
session_id	Ідентифікатор сесії	VARCHAR(100)

Дані про товари, які користувач додає до обраного зберігаються в таблиці обране. Це список товарів, які сподобалися користувачу сайту. Ця інформація також потрібна для подальшого навчання системи рекомендацій (див. табл. 3.13).

Таблиця 3.13

Атрибути таблиці обране на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
user_id	Зовнішній ключ на користувача	INT (FK users.id)
product_id	Зовнішній ключ на товар	INT (FK products.id)
added_at	Дата додавання	DATETIME

В таблиці кеш рекомендацій зберігаються дані про попередньо обчислені рекомендації для користувачів, що в роботі прискорюють відображення персоналізованих пропозицій. Тобто коли система буде вчитися, вона буде оброблювати великі масиви даних, які можуть мати однакові комбінації між користувачами та товарами. І доцільно було б використовувати готові обрахунки, які співпадають при певному наборі вхідних параметрів, ніж дублювати записи в таблиці бази даних знов (див. табл. 3.14).

Таблиця 3.14

Атрибути таблиці кеш рекомендацій на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
user_id	Зовнішній ключ на користувача	INT (FK users.id)

product_id	Зовнішній ключ на товар	INT (FK products.id)
score	Оцінка релевантності	DECIMAL(5,4)
algorithm_type	Тип алгоритму рекомендації	ENUM
explanation	Пояснення рекомендації	TEXT
generated_at	Дата генерації	DATETIME
expires_at	Дата закінчення актуальності	DATETIME

Для збереження статистики взаємодії користувача з рекомендаціями та рекомендованими товарами створена окрема таблиця кліки на рекомендації. В подальшому ці дані використовуються для оцінки ефективності алгоритмів (див. табл. 3.15).

Таблиця 3.15

Атрибути таблиці кліки на рекомендації на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
user_id	Зовнішній ключ на користувача	INT (FK users.id)
product_id	Зовнішній ключ на товар	INT (FK products.id)
recommendation_type	Тип рекомендації	ENUM
position	Позиція в списку рекомендацій	INT
clicked_at	Дата та час кліку	DATETIME

В таблиці схожість товарів зберігається інформація про попередньо обчислені коефіцієнти схожості між товарами. Це прискорює роботу алгоритмів рекомендацій. Це створено для того, щоб не дублювати вже розраховані показники при однакових вхідних наборах параметрів. Таким чином оптимізується процес запису в таблицю та уникається дублювання записів та перенавантаження бази даних (див. табл. 3.16).

Таблиця 3.16

Атрибути таблиці схожість товарів на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
product_id	Зовнішній ключ на товар	INT (FK products.id)

similar_product_id	Зовнішній ключ на схожий товар	INT (FK products.id)
similarity_score	Коефіцієнт схожості	DECIMAL(5,4)
similarity_type	Тип схожості	ENUM
calculated_at	Дата обчислення	DATETIME

В таблиці схожість користувачів зберігається інформація про коефіцієнти схожості між користувачами для колаборативної фільтрації. Ці дані дозволяють алгоритму знаходити користувачів зі схожими вподобаннями (див. табл. 3.17).

Таблиця 3.17

Атрибути таблиці схожість товарів на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
user_id	Зовнішній ключ на користувача	INT (FK users.id)
similar_user_id	Зовнішній ключ	INT (FK users.id)
similarity_score	Коефіцієнт схожості	DECIMAL(5,4)
calculated_at	Дата обчислення	DATETIME

Для того, щоб адміністратору системи було легко виявляти помилки чи несанкціоновані входи реалізовано систему легування. В таблиці системні логи зберігається журнал подій системи для моніторингу та діагностики. Тут система буде зберігати інформаційні повідомлення та попередження (див. табл. 3.18).

Таблиця 3.18

Атрибути таблиці системні логи на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
log_type	Тип логу	ENUM
component	Компонент системи	VARCHAR(100)
message	Текст повідомлення	TEXT

Таблиця метрики рекомендацій зберігає в собі щоденні показники ефективності алгоритмів рекомендацій. Ці дані використовуються для аналізу та оптимізації системи (див. табл. 3.19).

Таблиця 3.19

Атрибути таблиці метрики рекомендацій на логічному рівні

Назва атрибуту	Опис	Тип даних
id	Унікальний ідентифікатор запису	INT (PK)
metric_date	Дата метрики	DATE
algorithm_type	Тип алгоритму	ENUM
total_recommendations	Загальна кількість рекомендацій	INT
clicked_recommendations	Кількість кліків на рекомендації	INT
ctr	Click-Through Rate	DECIMAL(5,4)
avg_score	Середня оцінка рекомендацій	DECIMAL(5,4)
coverage	Покриття каталогу	DECIMAL(5,4)
diversity	Різноманітність рекомендацій	DECIMAL(5,4)
calculated_at	Дата обчислення	DATETIME

Таким чином, сутності, виявлені під час концептуального моделювання, були сформовані у вигляді таблиць на логічному рівні проектування бази даних, де кожна таблиця зберігає свій набір даних.

3.3.3 Фізична модель

Отримавши логічну модель даних, тепер є набір атрибутів для кожної таблиці. Для розгортання бази даних створюється фізична модель бази даних. За допомогою програмного забезпечення MySQL WorkBench було створено таблиці та встановлені зв’язки між ними. В налаштуваннях таблиці були обрані відповідні до розробленої логічної моделі типи даних для кожного з атрибутів. Зв’язки встановлювалися згідно з правилами нормалізації моделі бази даних. Створена модель приведена до 3НФ та має закінчений вигляд (див. рис. 3.9).

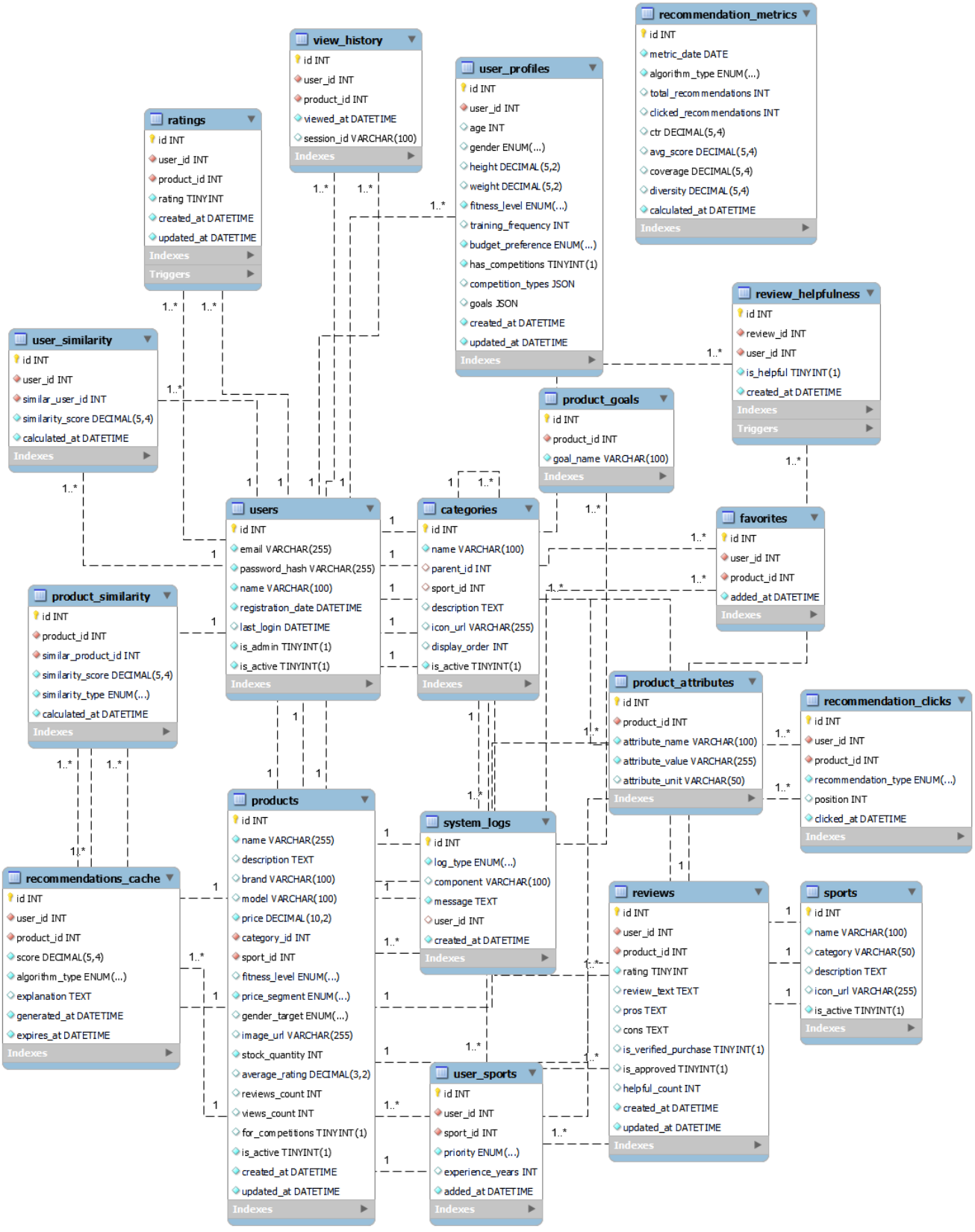


Рис. 3.9 ER діаграма фізичної моделі даних для системи рекомендацій

Тепер, використовуючи програмний засіб MySQL WorkBench було отримано SQL скрипт для розгортання фізичної моделі бази даних в СКБД.

3.4 Проєктування та реалізація алгоритму рекомендацій

3.4.1 Математична модель

Всі рекомендації в системі базуються на співвідношені характеристик користувача і характеристик товарів. Профіль користувача і картка спортивного спорядження представлені в системі як багатовимірні вектори у спільному просторі ознак (формула 5.1).

$$U = (S, L, B, G, F, C, P) \tag{5.1}$$

де S - множина видів спорту,

L - рівень підготовки,

B - бюджетні преференції,

G - множина цілей тренувань,

F - частота тренувань,

C - участь у змаганнях,

P - фізичні параметри.

Товар аналогічно в системі представлений як багатовимірний вектор, який залежить від атрибутів (формула 5.2).

$$P = (s, l, b, g, r, t, c) \tag{5.2}$$

де s — вид спорту,

l — рівень підготовки,

b — ціновий сегмент,

g — множина цілей, для яких призначений товар,

r — рейтинг,

t — тип товару,

c — категорія.

Для оцінки схожості вподобань користувача з атрибутам товару використовується зважена формула оцінки релевантності товару для користувача, що враховує шість основних копонентів, які утворюють Content-Based метод оцінки (формула 5.3).

$$Score_{CB}(U,P) = \omega_s * S_{sport}(U,P) + \omega_l * S_{level}(U,P) + \omega_b * S_{budget}(U,P) + \omega_g * S_{goals}(U,P) + \omega_c * S_{comp}(U,P) + \omega_r * S_{rating}(P) \quad (5.3)$$

де $\omega_s -$ відповідність виду спорту = 0,40

$\omega_l -$ відповідність рівню фізичної підготовки клієнта $= 0{,}25$

$\omega_b -$ відповідність бюджету клієнта $= 0{,}15$ $\omega_g -$ відповідність кількісті аналогічних покупок $= 0{,}10$ $\omega_c -$ відповідність відношення до змагань $= 0{,}05$ $\omega_r -$ відповідність рейтенгу $= 0{,}05$

Ці компоненти відображають їх відносну важливість у загальній оцінці. Схожість між товарами формує метрику рекомендацій, яка показує схожі товари користувачу на основі його переглядів, яка формується на основі зважених метрик (формулу 5.4).

$$\text{Similarity}(P_1,P_2) = 0.4 \cdot \delta(P_1.\text{s},P_2.\text{s}) + 0.3 \cdot \delta(P_1.\text{c},P_2.\text{c}) + 0.15 \cdot \text{S}_{\text{level}}(P_1.\text{l},P_2.\text{l}) + 0.1 \cdot \delta(P_1.\text{b},P_2.\text{b}) + 0.05 \cdot \delta(P_1.\text{brand},P_2.\text{brand}) \quad (5.4)$$

де $\delta(a,b)$ - функція Кронекера.

Для оцінки схожості користувачів між собою розроблено комплексну метрику схожості з урахуванням трьох компонентів (формула 5.5)

$$\text{Similarity}(U_1, U_2) = 0.4 \cdot \text{S}_{\text{sports}}(U_1, U_2) + 0.3 * \text{S}_{\text{profile}}(U_1, U_2)$$
$$+0.3 * \text{S}_{\text{rating}}(U_1, U_2) \quad (5.5)$$

Схожість за видами спорту використовує коефіцієнт Жаккара. Таким чином користувачу рекомендуються товари зі схожого виду спорту за категорією (формула 5.6).

$$S_{sports}(U_1, U_2) = \frac{|U_1 * S \cap U_2 * S|}{|U_1 * S \cup U_2 * S|} \quad (5.6)$$

Схожість оцінок товарів $S_{raiting}(U_1, U_2)$ базується на кореляції Пірсона для спільно оцінених товарів (формула 5.7).

$$S_{raiting}(U_1, U_2) = \frac{\sum_{p \epsilon P_{common}} (R_{1,p} - R_1)(R_{2,p} - R_2)}{\sqrt{\sum_{p \epsilon P_{common}} (R_{1,p} - R_1)^2} * \sqrt{\sum_{p \epsilon P_{common}} (R_{2,p} - R_2)^2}} \quad (5.7)$$

де $P_{common} -$ множина товарів, оцінених обома користувачами,
$R_{i,p} -$ оцінка товару p користувачем i
$R_i -$ середня оцінка користувача i

Прогнозування оцінки у колабораційному фільтруванні для товару p який користувач U ще не оцінив, розраховується як зважене середнє оцінок схожих користувачів (формула 5.8).

$$R_{u,p} = \frac{\sum_{U \in N(u)} Similarity\ (u,v) * R_{v,p}}{\sum_{U \in N(u)} Similarity\ (u,v)} \quad (5.8)$$

де N(u) - множина k найбільш схожих користувачів, які оцінили товар p.

Відповідно до цього, нормалізована оцінка для формування рекомендацій описується формулою 5.9, що проводить прогнозовану оцінку до діапазону [0, 1].

$$SCORE_{CF}(u,p) = \frac{R_{u,p}}{5} \quad (5.9)$$

Гібрідна оцінка є фінальною оцінкою релевантності товару в гібридній системі, що формується як зважена комбінація оцінок з обох методів (формула 5.10).

$$SCORE_{hybrid}(U,P) = \omega_{CB} * SCORE_{CB}(U,P) + \omega_{CF} * SCORE_{CF}(U,P) \quad (5.10)$$

Така адаптація ваг дозволяє системі покладатися більше на контентну фільтрацію для нових користувачів з малою кількістю оцінок, і більше на колаборативну фільтрацію для досвідчених користувачів з багатою історією взаємодій.

3.4.2. Реалізація алгоритму

Система рекомендацій спортивного спорядження працює на основі гібридного підходу. Цей підхід поєднує контентний підхід та колаборативний підхід, що компенсує недоліки кожного з двух методів.

Для того, щоб оптимізувати алгоритм рекомендацій, були створені таблиці, які збирають кеш про попередньо розраховані рекомендації, які були виконані за минули добу. Схожість товарів та користувачів заздалегідь розраховується та зберігається в базі даних. Завдяки цьому не потрібно робити додаткових обчислень.

Контентний підхід аналізує характеристики товарів та профіль користувача та знаходить певні схожості з основними показниками (див. табл. 3.20).

Таблиця 3.20

Компоненти оцінки схожості в контентному підході рекомендування

Компонент	Вага	Опис
Відповідність виду спорту	40%	Співпадіння з видами спорту користувача
Рівень підготовки	25%	Відповідність рівня спортивної підготовки користувача та товару
Бюджет	15%	Співпадіння цінового сегменту
Цілі тренувань	10%	Перетин цілей користувача та товару
Участь у змаганнях	5%	Співпадіння параметру участі у змаганнях
Рейтинг товару	5%	Нормалізований рейтинг товару

```
def _calculate_similarity_score(self, product):
score = 0.0
if product.sport_id in self.user_sports:
primary_sports = [us.sport_id for us in
self.user.sports.filter_by(priority='primary').all()]
if product.sport_id in primary_sports:
score += 0.40
else:
score += 0.30
score += self._fitness_level_score(product) * 0.25
score += self._budget_score(product) * 0.15
score += self._goals_score(product) * 0.10
if self.profile.has_competitions and product.for_competitions:
score += 0.05
if product.average_rating > 0:
score += (float(product.average_rating) / 5.0) * 0.05
return round(score, 4)
```

Колаборативний під’їзд базується на тому, що він аланілзує схожості різних користувачів, а потім рекомендує групі користувачів одні й ті ж самі товари, думаючи, що інтереси у них мають бути спільні. Розрахунки методи виконуються за трьома критеріями див. табл. 3.21).

Таблиця 3.21

Компоненти схожості користувачів у колаборативному підході

Компонент	Вага	Метод розрахунку
Схожість видів спорту	40%	Метод Жакарда на множинах sport_id
Схожість профілів	30%	Порівняння fitness_level, budget, goals, training_frequency
Схожість оцінок	30%	Pearson correlation на спільно оцінених товарах

```
def _ratings_similarity(self, user1_id, user2_id):
user1_ratings = db.session.query(Rating.product_id,
Rating.rating) \ .filter_by(user_id=user1_id).all()
user2_ratings = db.session.query(Rating.product_id,
Rating.rating) \ .filter_by(user_id=user2_id).all()
user1_dict = {product_id: rating for product_id, rating in
user1_ratings}
user2_dict = {product_id: rating for product_id, rating in
user2_ratings}
common_products = set(user1_dict.keys()) &
set(user2_dict.keys())
if len(common_products) < 2:
return 0.0
ratings1 = [user1_dict[pid] for pid in common_products]
ratings2 = [user2_dict[pid] for pid in common_products]
mean1 = np.mean(ratings1)
mean2 = np.mean(ratings2)
diff1 = [r - mean1 for r in ratings1]
diff2 = [r - mean2 for r in ratings2]
numerator = sum([d1 * d2 for d1, d2 in zip(diff1, diff2)])
denominator = np.sqrt(sum([d ** 2 for d in diff1])) * \
np.sqrt(sum([d ** 2 for d in diff2]))
if denominator == 0:
return 0.0
correlation = numerator / denominator
return (correlation + 1) / 2
```

Алгоритм програми реалізовано таким чином, щоб пройтись по всім підходам та обрати найкращий варіант. Тобто алгоритм пропонує кілька стратегій генерації рекомендації. Після чого формуються адаптивні ваги гібридного алгоритму, які основані на кількості оцінок користувача. Система фіксує клік на рекомендовані товари з метаданими про тип алгоритму та позицію в списку, що забезпечує можливість аналізу ефективності різних підходів (див. табл. 3.22).

Таблиця 3.22

Адаптивні ваги гібридного підходу

Кількість оцінок користувача	Content-based	Collaborative	Обґрунтування
< 5	80%	20%	Недостатньо даних для collaborative
5-14	60%	40%	Збалансований підхід
≥ 15	40%	60%	Достатньо даних для точного collaborative

```
def get_weighted_recommendations(self, limit=10):
from app.models import Rating
user_ratings_count                                                    =
Rating.query.filter_by(user_id=self.user_id).count()
if user_ratings_count < 5:
content_weight = 0.8
collaborative_weight = 0.2
elif user_ratings_count < 15:
content_weight = 0.6
collaborative_weight = 0.4
else:
content_weight = 0.4
collaborative_weight = 0.6
return self.get_recommendations(limit, content_weight,
collaborative_weight)
```

Різноманітні рекомендації формуються шляхом вибірки з потрійного набору кандидатів з контролем розподілу по категоріях та брендах для уникнення одноманітності.

3.4.3. Метрики оцінювання

Метрики оцінювання формуються класом RecomendationMetrics. Тут реалізовано набір стандартних метрик для рекомендаційної системи. Точність передбачення рейтингів вимірюється через RMSE та MAE. Вони обчислюються як середньоквадратичне та середня абсолютна помилка між реальними та передбаченими оцінками юзерів. Ці метрики дають можливість оцінити чи точно система рекомендує, чи шось не то каже.

```
@staticmethod
def calculate_rmse(true_ratings, predicted_ratings):
```

```
    if len(true_ratings) == 0:
    return 0.0
    return                    np.sqrt(mean_squared_error(true_ratings,
predicted_ratings))
    @staticmethod
    def calculate_mae(true_ratings, predicted_ratings):
    if len(true_ratings) == 0:
    return 0.0
    return mean_absolute_error(true_ratings, predicted_ratings)
```

Релевантність рекомендацій оцінюється метриками Precision@K, Recall@K та F1-Score. Precision@K визначає частку релевантних товарів серед топ-K рекомендацій, Recall@K – частку знайдених релевантних товарів від усіх релевантних, F1-Score – гармонічне середнє між ними. Релевантними вважаються товари з оцінкою користувача ≥4 зірки, що відповідає високому ступеню задоволеності. Ці метрики є ключовими для оцінки того, наскільки рекомендована система пропонує товари, які дійсно цікаві користувачу.

```
    @staticmethod
    def calculate_precision_at_k(recommended_items, relevant_items,
k):
    if k == 0:
    return 0.0
    recommended_at_k = set(recommended_items[:k])
    relevant_set = set(relevant_items)
    hits = len(recommended_at_k & relevant_set)
    return hits / k
    @staticmethod
    def calculate_recall_at_k(recommended_items, relevant_items, k):
    if len(relevant_items) == 0:
    return 0.0
    recommended_at_k = set(recommended_items[:k])
    relevant_set = set(relevant_items)
    hits = len(recommended_at_k & relevant_set)
    return hits / len(relevant_set)
    @staticmethod
    def calculate_f1_score(precision, recall):
    if precision + recall == 0:
    return 0.0
    return 2 * (precision * recall) / (precision + recall)
```

Система метрик розроблена з урахуванням специфіки предметної області спортивного спорядження, де критично важлива не лише точність рекомендацій, але й їхня різноманітність та покриття каталогу.

3.5 Реалізація програмного забезпечення

3.5.1 Налаштування середовища розробки

Для розробки був інстальований програмний продукт PyCharm, який є гарним редактором коду для створення проектів на Python. В цьому середовищі розробки встроєний вже свій термінал, для роботи з пакетами та бібліотеками. Жодної сторонньої утілити не треба більше. Віртуальне середовище було створено через вбудовані можливості venv (див. рис. 3.10).

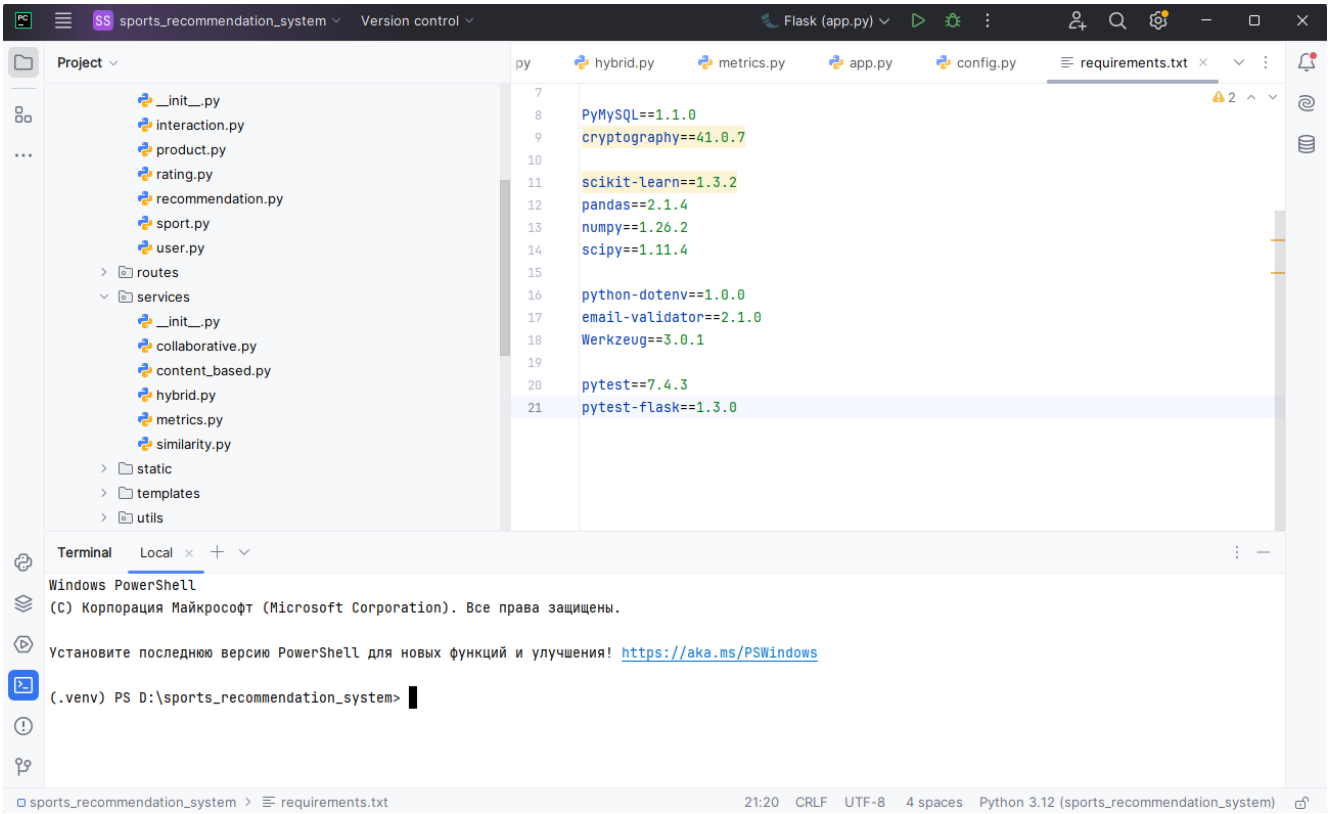


Рис. 3.10 Підготовка середовища розробки PyCharm

Для зручності розгортання проекту в подальшому в окремому файлі була визачена структура залежностей проекту, тобто прописані всі залежності, які

потрібно інсталювати. Для того, щоб це зробити однією командою, а не інстальовувати кожну залежність окремо, потрібно виконати команду в терміналі.

```
pip install -r requirements.txt
```

В файлі із залежностями прописані абсолютно всі залежності, такі як Flask-SQLAlchemy 3.1.1 для ORM, PyMySQL 1.1.0 як MySQL-драйвер для SQLAlchemy. Для машинного навчання підключені scikit-learn 1.3.2, pandas 2.1.4 та numpy 1.26.2, що забезпечують розрахунок схожості та метрик. Flask-Login 0.6.3 відповідає за автентифікацію користувачів, Flask-Migrate 4.0.5 – за міграції бази даних, python-dotenv 1.0.0 – за керування змінними оточення.

```
Flask==3.0.0
Flask-SQLAlchemy==3.1.1
Flask-Migrate==4.0.5
Flask-Login==0.6.3
Flask-WTF==1.2.1
WTForms==3.1.1
PyMySQL==1.1.0
cryptography==41.0.7
scikit-learn==1.3.2
pandas==2.1.4
numpy==1.26.2
scipy==1.11.4
python-dotenv==1.0.0
email-validator==2.1.0
Werkzeug==3.0.1
pytest==7.4.3
pytest-flask==1.3.0
```

PyCharm налаштовується з Flask run configuration, що автоматично виявляє структуру проєкту та забезпечує hot-reload при змінах коду. Відладчик PyCharm дозволяє встановлювати breakpoint'и безпосередньо в коді рекомендаційних алгоритмів для покрокового аналізу розрахунків схожості.

Конфігурація підключення до бази даних прописана в окремому файлі .env, в якому прописані всі дані для коректу.

```
SECRET_KEY=123456
```

```
FLASK_ENV=development
DB_HOST=127.0.0.1
DB_PORT=3306
DB_NAME=sports_recommendation_system
DB_USER=root
DB_PASSWORD=
RECOMMENDATION_CACHE_HOURS=24
MIN_SIMILARITY_THRESHOLD=0.3
DEFAULT_RECOMMENDATIONS_LIMIT=10
```

Таким чином організована автоматична можливість розгортання проекту на будь-якому локальному комп'ютері без загроз забути щось імпортувати.

3.5.2. Серверна частина на Flask

Конфігурація додатку реалізована через класи Config з підтримкою різних середовищ. Базовий клас Config містить загальні параметри, включаючи підключення до MySQL через PyMySQL драйвер. Всі моделі системи реалізовані в папці models (див. рис. 3.11).

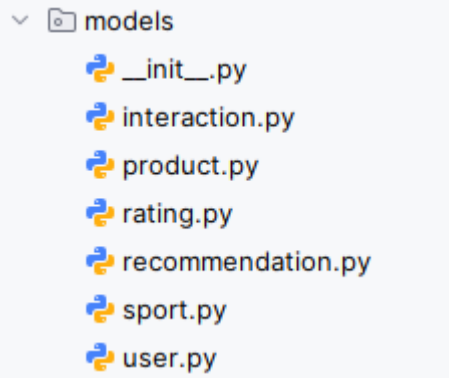


Рис. 3.11 Створення моделей системи

Метод формування SQLALCHEMY_DATABASE_URI враховує можливість порожнього пароля для локальної розробки з WampServer. Параметр SQLALCHEMY_ECHO=True увімкнений для режиму розробки, що дозволяє відстежувати SQL-запити в консолі PyCharm. Клас DevelopmentConfig активує режим відладки, ProductionConfig вимикає виведення SQL та DEBUG для

продуктивності. Всі сервіси для реалізації алгоритмів рекомендації реалізовані в папці services (див. рис. 3.12).

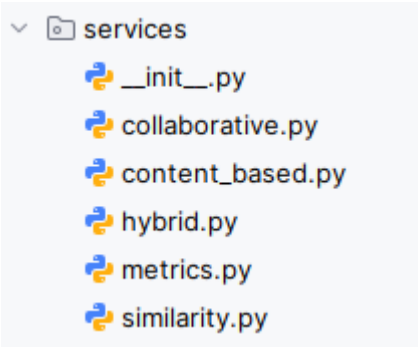


Рис. 3.12 Створення сервісів системи

Ініціалізація Flask додатку відбувається через патерн Application Factory у модулі app/init.py. Функція create_app створює екземпляр додатку, завантажує конфігурацію з config.py, ініціалізує розширення (SQLAlchemy, LoginManager, Migrate) та реєструє Blueprint'и для маршрутизації. SQLAlchemy налаштована для роботи з MySQL через PyMySQL драйвер, що забезпечує сумісність з Python 3.9+. Login Manager налаштований з перенаправленням на сторінку входу для неавторизованих користувачів. Flask-Migrate інтегрований для керування міграціями структури БД.

API рекомендацій реалізовано через окремий Blueprint з RESTful ендпоінтами. Параметр algorithm у query string дозволяє явно вибрати тип алгоритму. Всі відповіді форматуються у JSON з полями success, data та опціональним error.

Точка входу додатку визначена у файлі run.py, який створює екземпляр Flask через create_app та запускає сервер розробки. Shell context додає основні моделі до Flask shell для зручного тестування через консоль PyCharm.

3.5.3 Реалізація бази даних MySQL

Створена фізична модель в програмі MySQL WorkBench має можливість сгенерувати готовий SQL скрипт для розгортання бази даних в СКБД MySQL. Із

запущеним WampServer було виконано команду Forward Engeneiring та отримано набір SQL скриптів для створення таблиць бази даних.

Перед додаванням таблиць в базу даних була створена база даних за допомогою виконаного SQL скрипта. Для цього в СКБД був обраний режим роботи з SQL та виконаний скрипт

```
CREATE DATABASE sports_recommendation_system;
```

Після цього був виконаний скрипт для створеня таблиць бази даних, який представлений в додатку Б. В результаті в СКБД було створено всі таблиці бази даних, які готові для подальшого заповнення (див. рис. 3.13)

Рис. 3.13 Результат створення бази даних

Таким чином було створено базу даних в локальному сервері, який працює через програму WampServer. Коли локльний сервер на комп'ютері буде включений, то програма зможе взаємодіяти із базою даних.

3.5.4 Клієнтська частина та інтеграція

Клієнтська частина додатку реалізована за допомогою шаблонів. Створено базовий шаблон, який реалізує загальну розмітку, яка притаманна всім сторінкам веб додатку. А також реалізовані окремі шаблони, які завантажуються в залежності від маршрута. Всі ці шаблони зберігаються у папці tampletes (див. рис. 3.14).

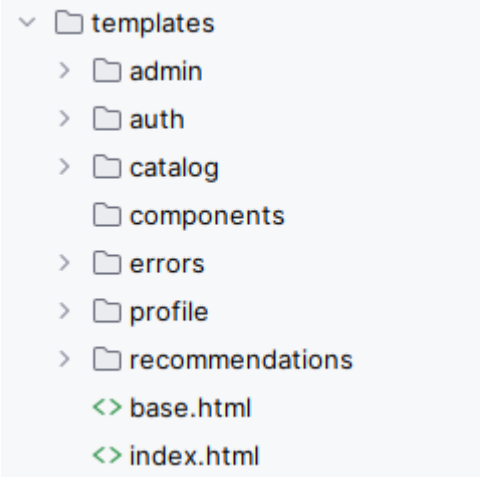


Рис. 3.14 Структура збереження шаблонів

Інтеграція клієнтської та серверної частин забезпечує безшовну взаємодію через RESTful API. CSRF-токени передаються в заголовках запитів для захисту від міжсайтової підробки запитів. Axios налаштований з базовим URL та перехоплювачами для автоматичного додавання токенів автентифікації. LocalStorage використовується для кешування налаштувань користувача (вибраний алгоритм, кількість товарів). Progressive Enhancement застосовується для підтримки базового функціоналу без JavaScript. Responsive дизайн адаптує інтерфейс для мобільних пристроїв, планшетів та десктопів з breakpoint'ами Bootstrap.

```
axios.defaults.baseURL = window.location.origin;
axios.defaults.headers.common['Content-Type'] = 'application/json';

const csrfToken = document.querySelector('meta[name="csrf-token"]');
if (csrfToken) {
    axios.defaults.headers.common['X-CSRFToken'] =
csrfToken.content;
}
```

```
axios.interceptors.request.use(
    config => {
        console.log(`[API] ${config.method.toUpperCase()}
${config.url}`);
        return config;
    },
    error => {
        console.error('[API] Помилка запиту:', error);
        return Promise.reject(error);
    }
);
```

3.5.5 Інструкція користувача

Після запуску програмного продукту в терміналі PyCharm з'являється інформація про вдале підключення до бази даних та апуску локального сервера, після чого веб додаток стартуєна порті 5000 (див. рис. 3.15).

Рис. 3.15 Запуск проєкта та вивід інформації в терміналі.

На головній сторінкі відображаються цікаві товари магазину та призив зареєструватися чи авторизуватися, щоб отримати всі можливосі.

На головній сторінці присутня форма для пошуку на сайті, посилання на авторизації, категорії товарів по видам спорту та популярні товари системи (див. рис. 3.16).

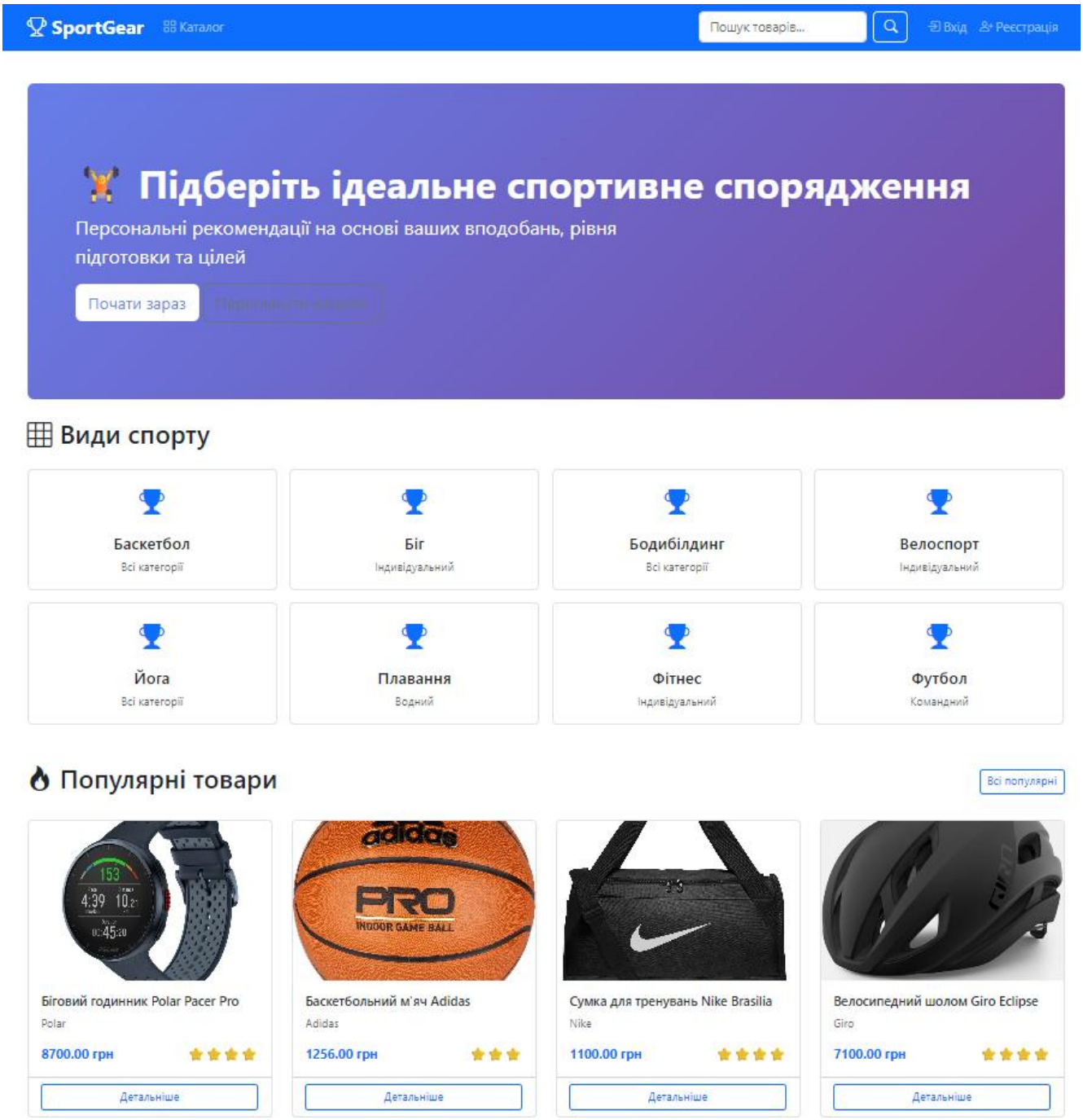


Рис. 3.16 Головна сторінка веб-додатку

Для реєстрації у веб-додатку потрібно заповнити форму реєстрації та придумати пароль для подальшого входу у систему (див. рис. 3.17).

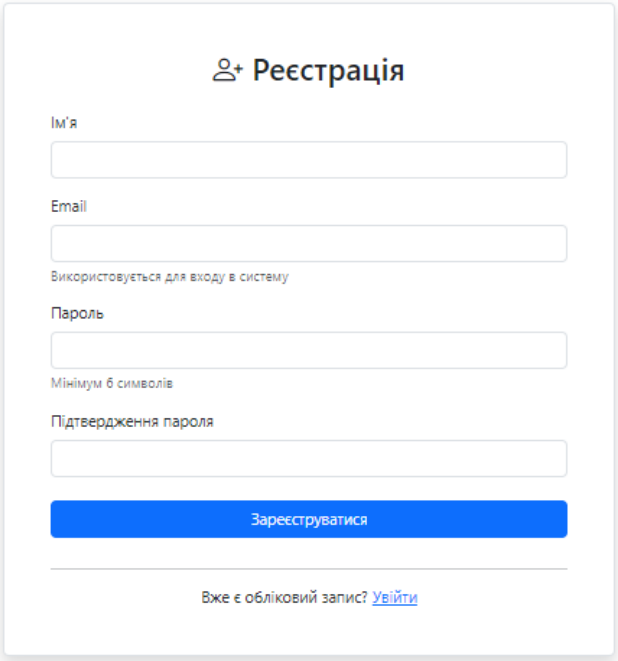


Рис. 3.17 Форма реєстрації у веб-додатку

Процес реєстрації проходить покроково. На першому кроці користувачу пропанується вибрати види спорту, якими він цікавиться (див. рис. 3.18).

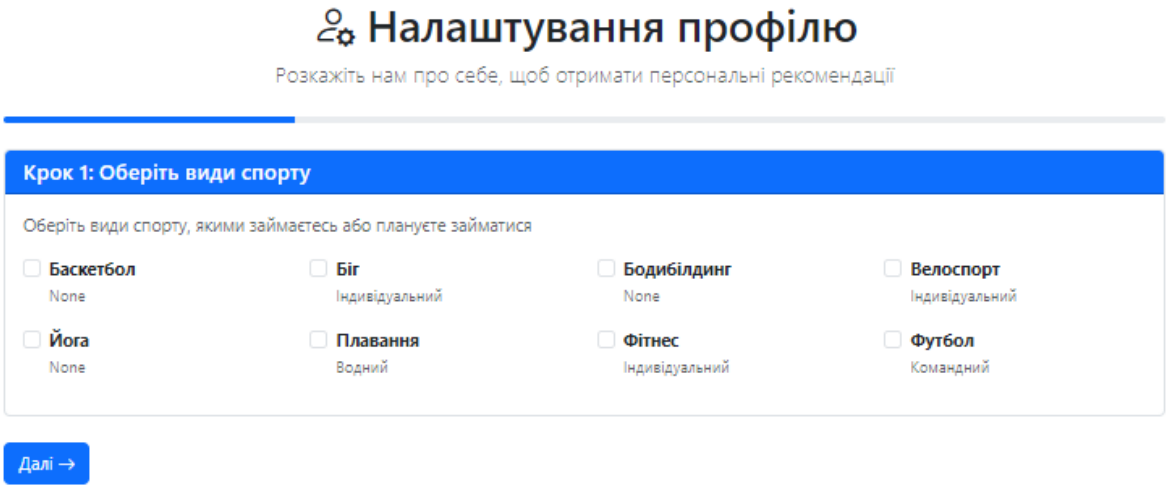


Рис. 3.18 Вибір видів спорту, якими цікавиться користувач

На другому кроці алаштування профілю, користувач вказує інформацію про свій рівень фізичної підготовки та вказує кількість занять спортом щотижня (див. рис. 3.19).

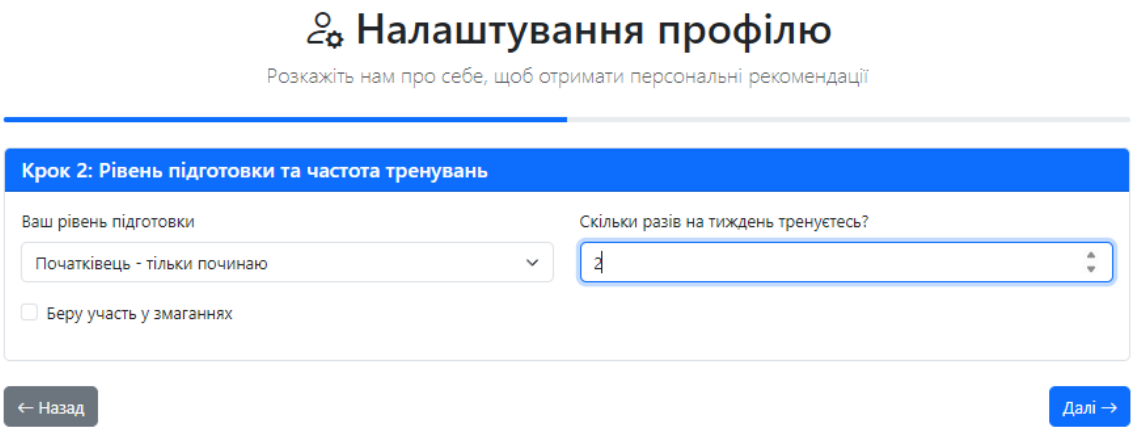


Рис. 3.19 Заповнення інформації про рівень фізичної підготовки

На третьому кроці заповнення анкети, користувач вказує мети заняття спортом та рівень бюджету, який він готовий витрачати на спортивне спорядження (див. рис. 3.20).

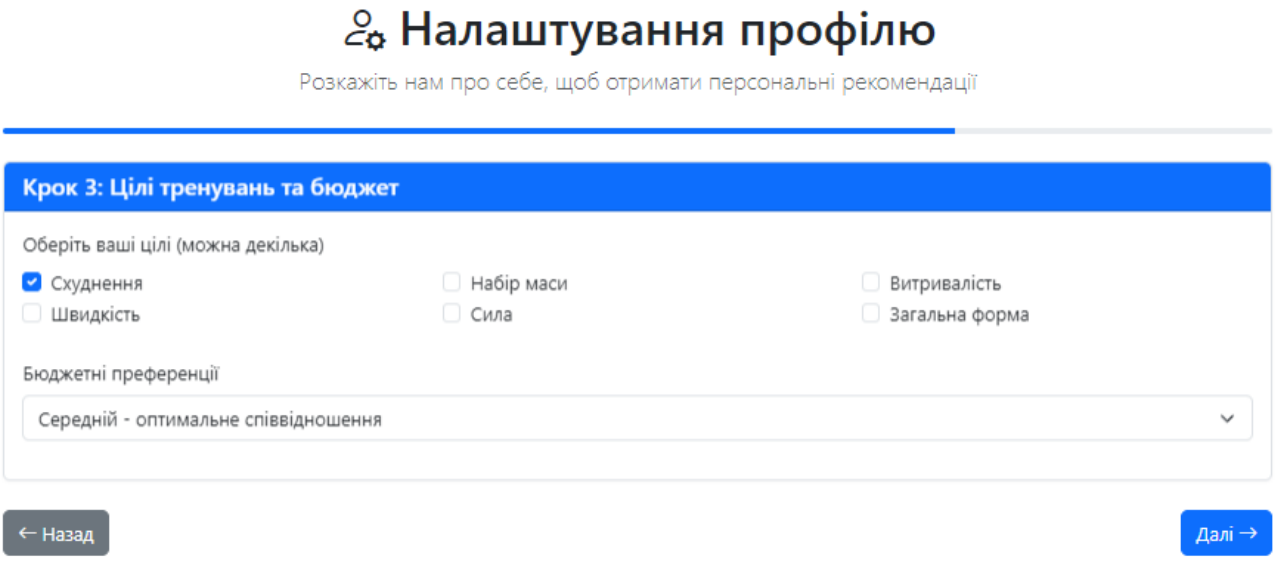


Рис. 3.20 Заповнення інформації про мети заняття спортом та бюджет

На четвертому кроці заповнення анкети, користувач вказує свої параметри, а саме зріст, вік, стать і вагу (див. рис. 3.21).

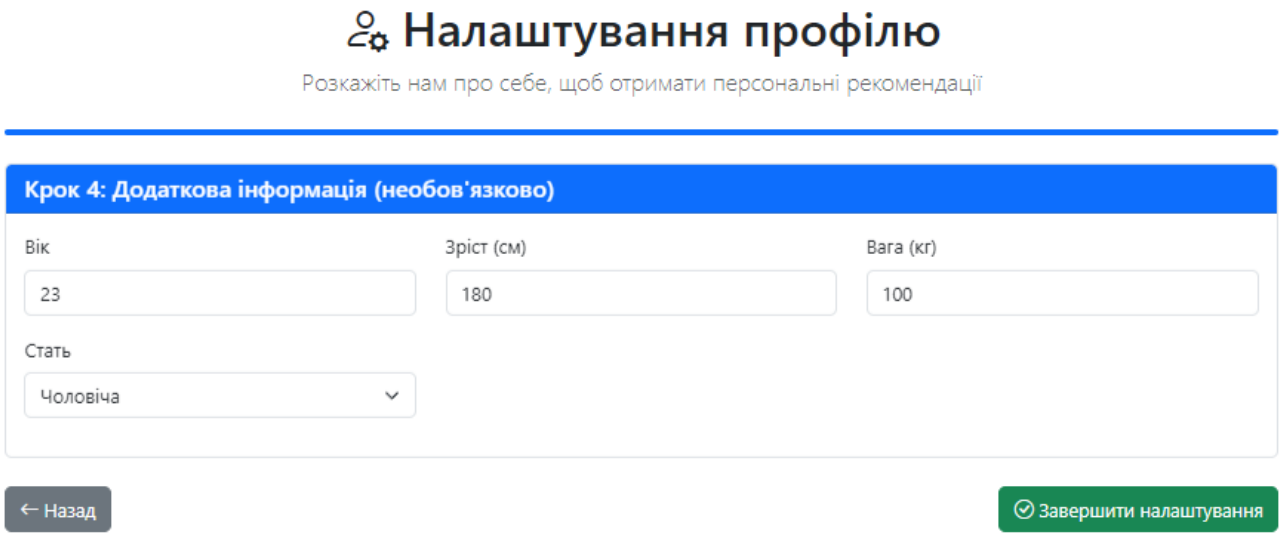


Рис. 3.21 Заповнення параметрів в анкеті користувача

Після реєстрації, користувачу відразу стає доступним весь функціонал системи. На основі анкети, яку користувач заповнював під час реєстрації, система рекомендує товар, який має йому сподобатися, базуючись на ціновому діапазоні та вподобаннях то виду спорту (див. рис. 3.22).

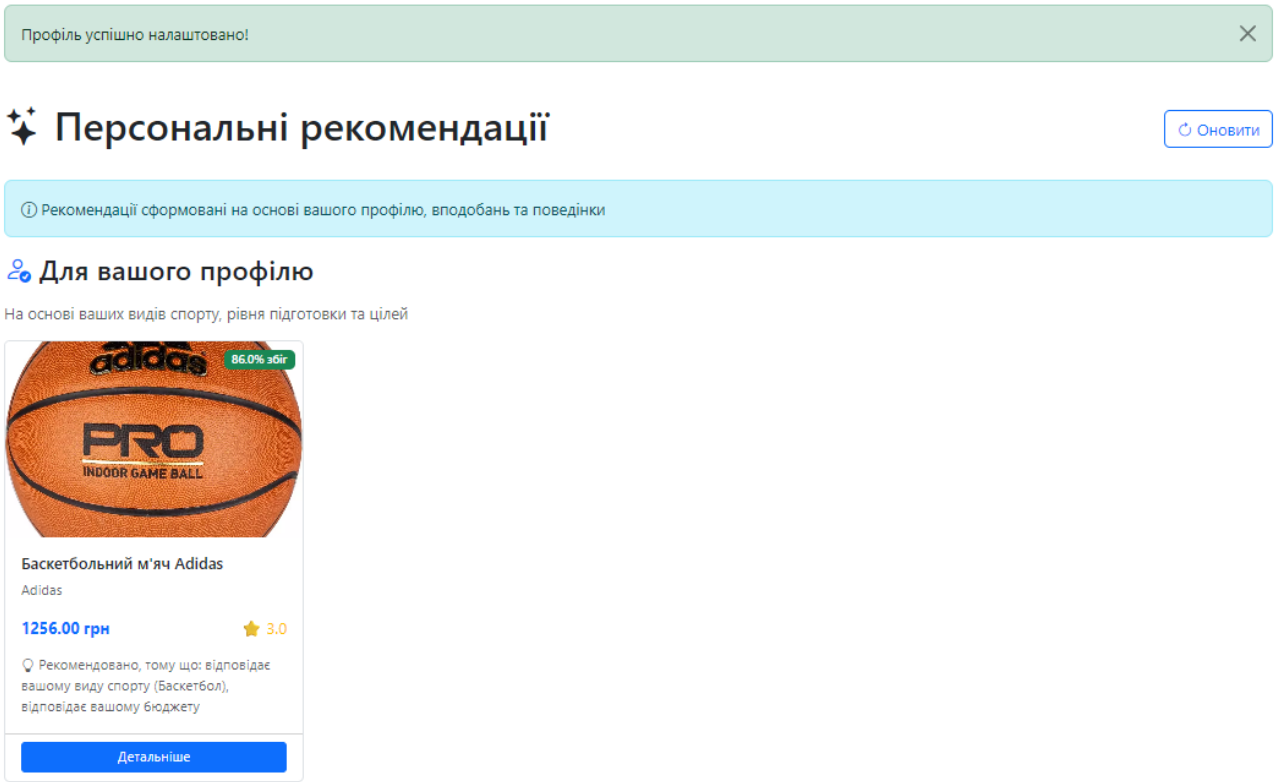


Рис. 3.22 Сформована рекомендація на основі заповненої анкети

Також система рекомендує товари користувачу, які цікавлять користувачів, зі схожими анкетними даними (див. рис. 3.23).

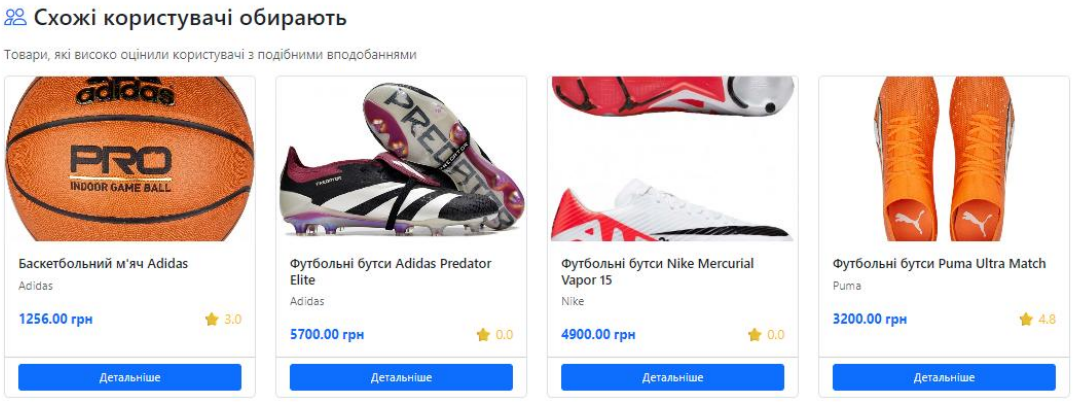


Рис. 3.23 Рекомендації на основі смаків схожих користувачів

Для зручності користування каталогом реалізована система фільтрації, яка налаштовуються атким чином, щоб користувач побачив саме ті товари, які його цікавлять (див. рис. 3.24).

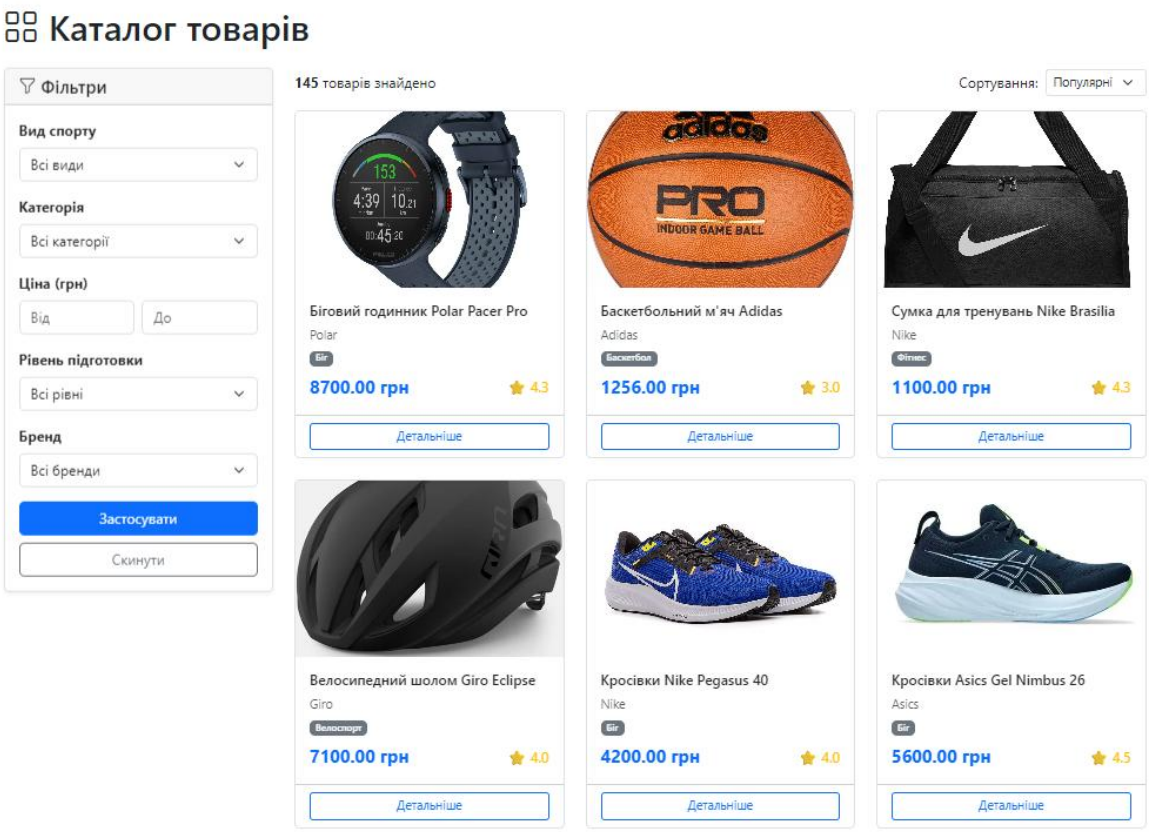


Рис. 3.24 Фільтрація каталогу товарів

Будь-який товар, який сподобався користувач, він може уважно роздивитися,перейшовши на сторінку детальної інформації, на якій вівдображаєтсья повна інформація, відгуки інших користувачів та оцінка на товар (див. рис. 3.25).

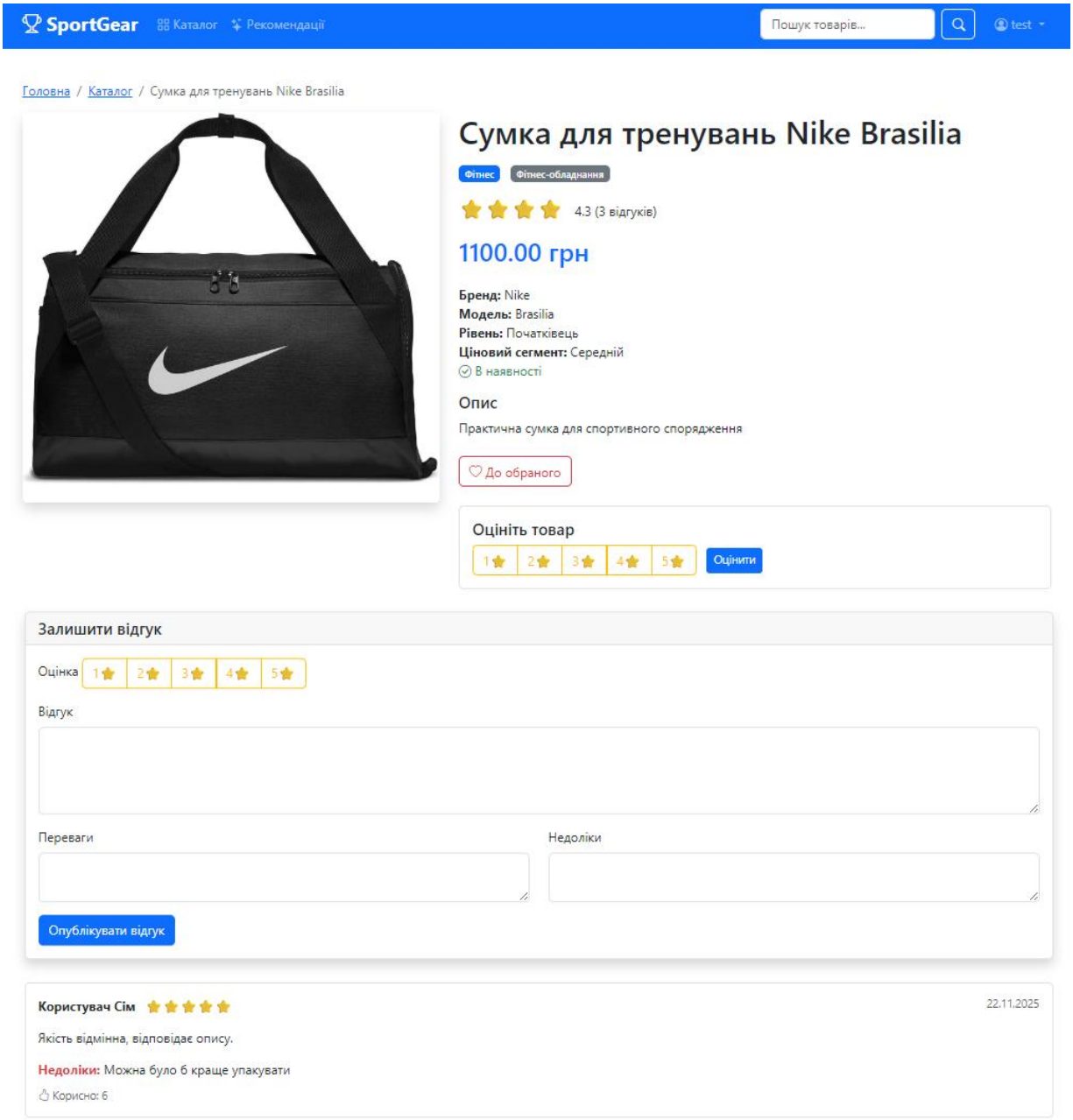


Рис. 3.25 Перегляд детальної інформації про товар та його оцінки

Сподобавшийся товар, користувач може додати до обраного. Для цього потрібно натиснути відповідну кнопку (див. рис. 3.26).

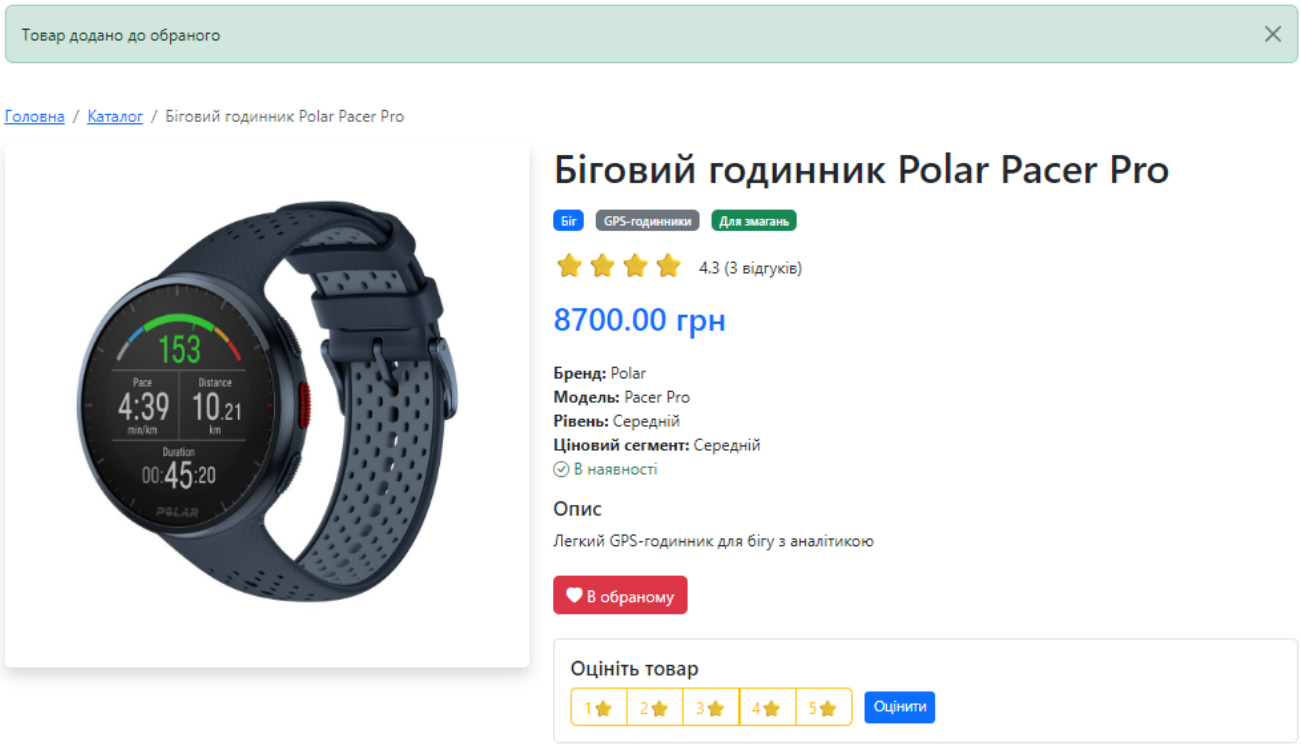


Рис. 3.26 Додавання товару до обраного

Після цього в розділі обране будуть відображатися ті товари, які сподобалися користувачу. В будь-який час він може видалити товар з обраного (див. рис. 3.27).

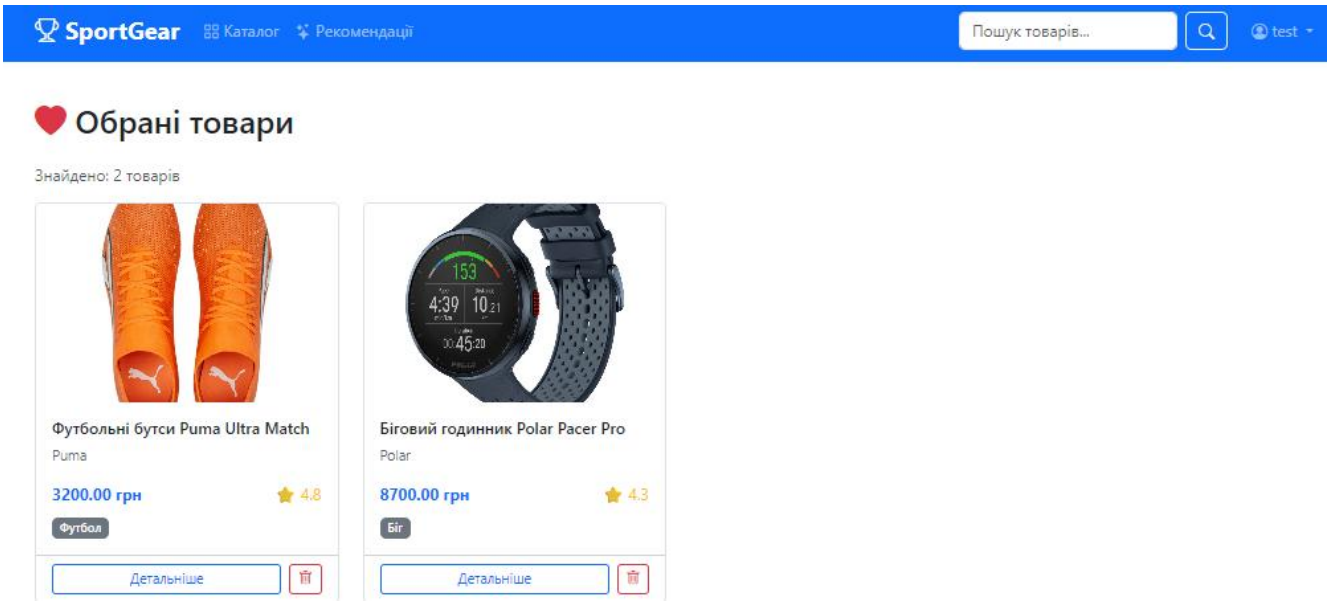


Рис. 3.27 Сторінка обраних товарів користувача

Якщо користувач переглядав товар, але він не може його повторно знайти, реалізована функція перегляду історії, де відображаються всі переглянуті товари користувачем (див. рис. 3.28).

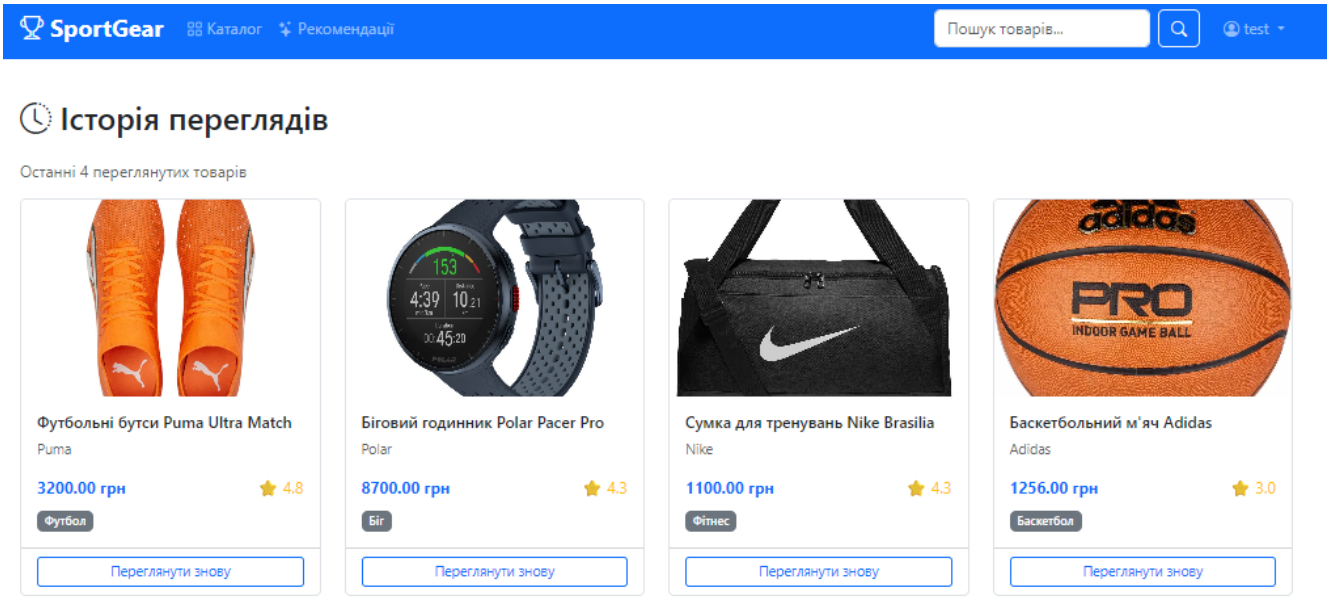


Рис. 3.28 Сторінка історії переглядів користувача

Також система рекомендує популярні товари, які переглядали інші користувачі і можуть зацікавити поточного користувача у розділі популярних категорій, вибраних під час реєстрації (див. рис. 3.29).

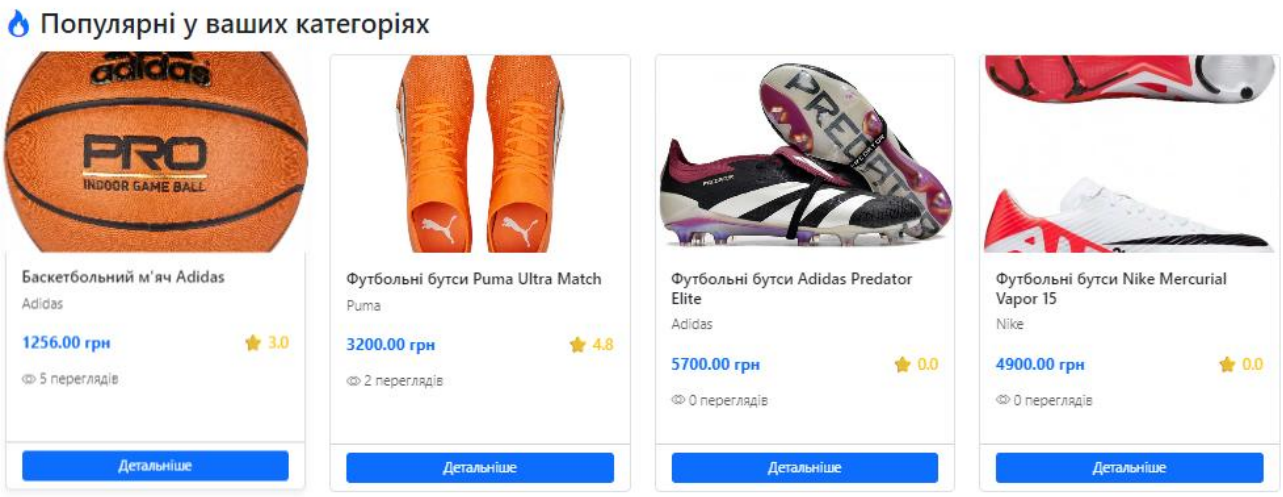


Рис. 3.29 Популярні товари в обраних категоріях

Для адміністратора реалізована своя панель керування системою. На сторінці адмін. панелі відображається швидка статистика та меню навігації (див. рис. 3.30).

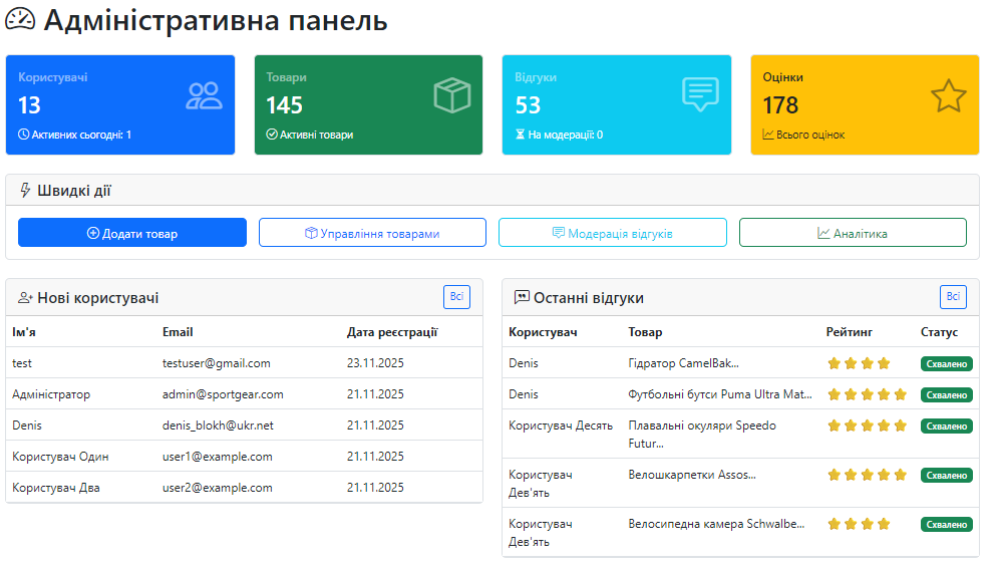


Рис. 3.30 Сторінка адміністратора

Адміністратор може додавати чи редагувати товари, які будуть відображатися у системі, вказуючи всі параметри та завантажуючи фото товару (див. рис. 3.31).

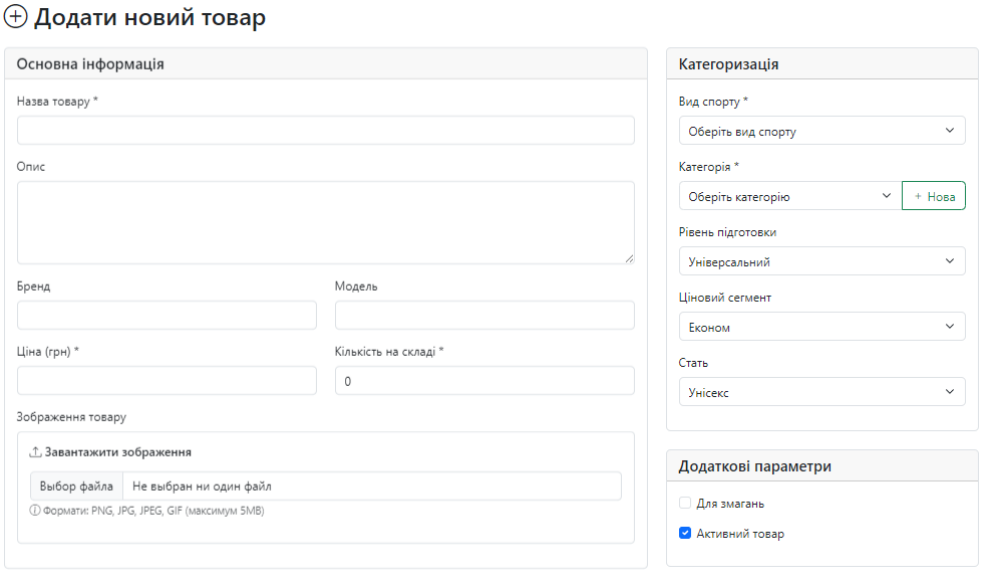


Рис. 3.31 Модуль додавання та редагування товарів

Зі сторінки керування товарами адміністратор може видаляти неактуальні товари чи переходити до їх релагування. Для зручності реалізовано пагінацію товарів по сторінкам (див. рис. 3.32).

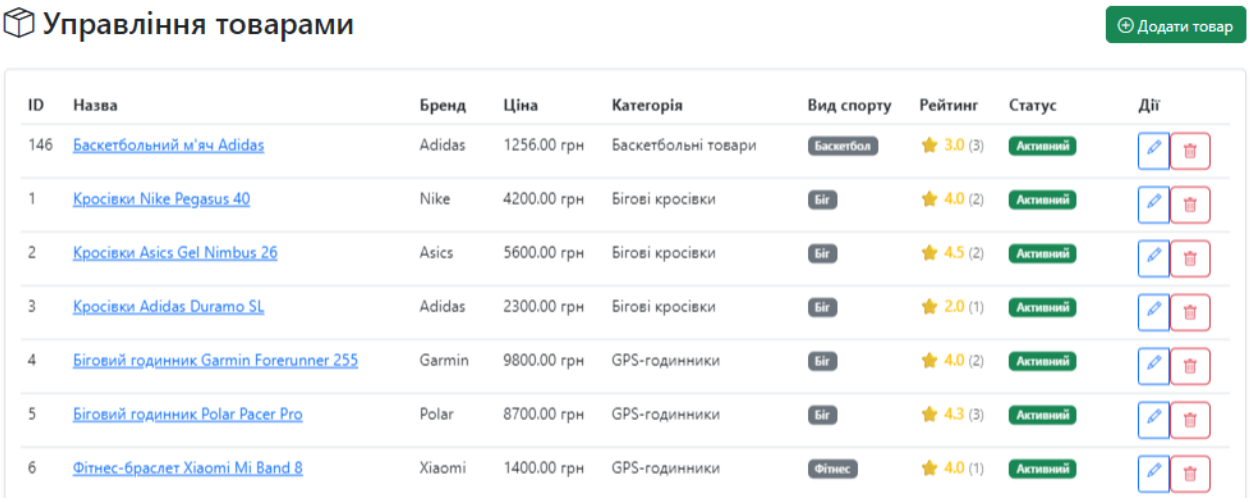


Рис. 3.32 Сторінка керування товарами

В розділі аналітики адміністратор може передивитися ефективність системи рекомендацій, як на рекомендації реагують користувачі (див. рис. 3.33).

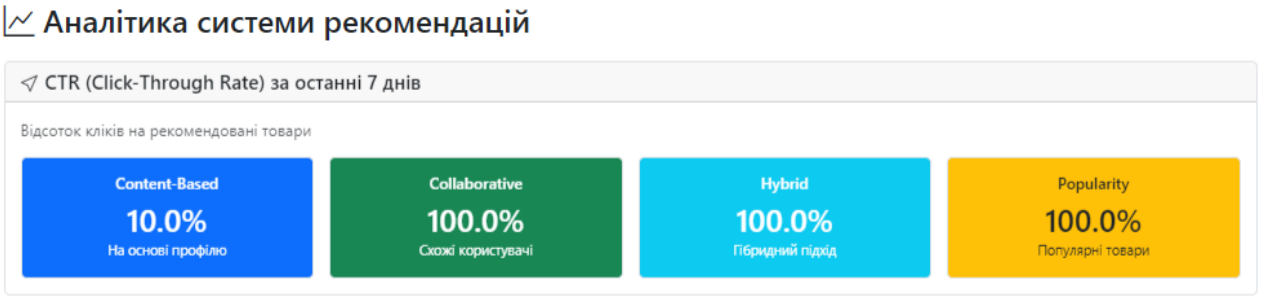


Рис. 3.33 Перегляд статистики ефективності системи рекомендацій

Таким чином в системі реалізована многорольова архітектура, яка дозволяє адміністратору керувати веб додатком, а користувачу ефективно обирати собі спортивне спорядження.

ВИСНОВКИ

В рамках кваліфікаційної роботи був виконаний великий обсяг труда, який всебічно охоплює всі процеси аналізу та розробки. Аналіз предметної галузі показав, що в світі поширюється тенденція філософії спорту і кількість людей, які активно займаються спортом стрімко збільшується. При цьому статистка говорить про те, що люди, які займаються спортом віддають перевагу придбанню якісного спортивного спорядження.

Тому актуальність системи, які автоматизує процес рекомендації товару та підбору спортивного спорядження є актуальний та обґрунтований реальними дослідженнями.

Процес рекомендації спортивного спорядження виконується в системі за декількома методами, результати попередньої рекомендації порівнюються відносно всіх методів та видається усереднений результат, який має найбільшу вірогідність успішної рекомендації для користувача. Основні фактори. На яких навчається модель системи пов’язані з поведінковими факторами користувача в системі та аналізу інформації, яку він вказує під час реєстрації. Даний проект може в подальшому слугувати як прототип для справжнього стартапу, який можливо, зможе конкурувати з такими гігантами в області спортивного спорядженя як Amaxon Sports. Якщо такий крок буде зроблений, то у проекта є всі шанси закріпитися на ринку маркетплейсів спортивного спорядження та отримати велику аудиторію.

Всі цілі проекту були досягнуті. Аналіз предметної галузі допоміг сформувати функціональні і не функціональні вимоги, які втілені в систему. Концептуальне моделювання предметної галузі слугувало базисом для формування сутностей системи та створення якісної бази даних, яка працює на СКБД MySQL. Розробка проєкта на мові програмування Python з використання фрейморка Flask відповідає сучасним тенденціям у виборі стеку розробки та допоможе мені в подальшому вказувати в резюе досвід роботи з цим технологічним стеком, який на сьогоднішній день користується добрим попитом.

Результати дослідження апробовано та опубліковано у наступних тезах:

1. Рекуц Д.Е., Трінтіна Н.А. Розробка інтелектуальної системи рекомендацій для підбору спортивного спорядження. V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник матеріалів. К.: ДУІКТ, 2025. С.91.

2. Рекуц Д.Е., Трінтіна Н.А. Застосування машинного навчання у створенні персоналізованих рекомендацій спортивного інвентарю. V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник матеріалів. К.: ДУІКТ, 2025. С.130.

ПЕРЕЛІК ПОСИЛАНЬ

1. Борецька Н. О. Фізична культура і спорт як один із пріоритетних напрямків державної політики. Молодий вчений. 2018. № 11 (2). С. 530–533.

2. Більшість українців не займається спортом, а 15,6 % опитаних роблять це регулярно – опитування Мінмолодьспорту. Інтерфакс-Україна. URL: https://ua.interfax.com.ua/news/sport/970624.html (дата звернення: 22.11.2025).

3. Про фізичну культуру і спорт : Закон України від 24.12.1993 № 3808-XII. URL: https://zakon.rada.gov.ua/laws/show/3808-12 (дата звернення: 22.11.2025).

4. Спорядження. Словник української мови : в 11 т. Київ : Наукова думка, 1970–1980. Т. 9. URL: http://www.inmo.org.ua/sum.html?wrd=Спорядження (дата звернення: 22.11.2025).

5. Іванов С. П. Ринок тренажерів з кожним роком збільшується. Бізнес. 2010. № 7. С. 27.

6. Daoud A. I., Geissler G. J., Wang F., Saretsky J., Daoud Y. A., Liebermann D. E. Foot strike and injury rates in endurance runners: a retrospective study. Medicine & Science in Sports & Exercise. 2012. Vol. 44, No. 7. P. 1325–1334.

7. Наконечний І. Ю., Циба Ю. Г. Основна проблематика менеджменту спортивних організацій в Україні. Інноваційна педагогіка. 2020. Вип. 22 (4). С. 247–251. URL: http://nbuv.gov.ua/UJRN/innped_2020_22(4)__52 (дата звернення: 22.11.2025).

8. Hoerzer S., von Tscharner V., Jacob C., Nigg B. M. Defining functional groups based on running kinematics using Self-Organizing Maps and Support Vector Machines. Journal of Biomechanics. 2015. Vol. 48, No. 10. P. 2072–2079.

9. Ворона В. В., Заяц С. В. Плавання : навч. посіб. для студ. закл. вищ. освіти спеціальності «017 Фізична культура і спорт». Суми : СумДПУ імені А. С. Макаренка, 2023. 168 с.

10. Горобей Н. В., Чмихал А. І., Терьохіна О. Л., Атаманюк С. І. Оздоровчі можливості вправ фітнес-аеробіки. Фізичне виховання та спорт у вищій школі. За

здоровий спосіб життя : зб. тез доп. Всеукр. наук.-практ. конф. (м. Запоріжжя, 15–16 жовтня 2009 р.). Запоріжжя : Запорізький НТУ, 2009. С. 8–9.

11. Chen D., Williams H., Garcia F. Advances in Shock-Absorbing Basketball Flooring Materials and Their Effect on Injury Prevention. International Journal of Sports Science. 2021. Vol. 29, No. 4. P. 125–140.

12. Casaca, Joaquim & Miguel, Luis. 2024. The Influence of Personalization on Consumer Satisfaction: Trends and Challenges. 10.4018/979-8-3693-3455-3.ch010. https://www.researchgate.net/publication/383006376_The_Influence_of_Personalization_on_Consumer_Satisfaction_Trends_and_Challenges

13. Abdelwahab, A., Sekiya, H., Matsuba, I., Horiuchi, Y., & Kuroiwa, S. Alleviating the sparsity problem of collaborative filtering using an efficient iterative clustered prediction technique. International Journal of Information Technology & Decision Making, 2012. 11(01), 33–53. https://doi.org/10.1142/S0219622012500022

14. Fayyaz Z.; Ebrahimian M.; Nawara D.; Ibrahim A.; Kashef R.Recommendation Systems: Algorithms, Challenges, Metrics, and BusinessOpportunities // Applied Sciences. 2020. Т. 10, № 21. С. 7748. URL:https://www.mdpi.com/2076-3417/10/21/7748 (дата звернення 02.11.2025). (mdpi.com)

15. Raza S. та ін. A Comprehensive Review of Recommender Systems:Transitioning from Theory to Practice // arXiv preprint arXiv:2407.13699.23.02.2025. URL: https://arxiv.org/abs/2407.13699 (дата звернення (02.11.2025)

16. Amazon Sports & Outdoor https://www.amazon.com/b/?_encoding=UTF8&ie=UTF8&node=10805321&ref_=cct_cg_F2MSOHOL25_1a1&pf_rd_p=688c6f74-698d-4fc5-8e01-85c95d584094&pf_rd_r=A6441GGMAAVPQ4PBR6D2

17. Dick's Sporting Goods : магазин спортивних товарів : офіційний вебсайт. URL: https://www.dickssportinggoods.com (дата звернення: 22.11.2025).

18. Roy D., Dutta M. A systematic review and research perspective on recommender systems. Journal of Big Data. 2022. Vol. 9, Art. 59. URL:

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00592-5 (дата звернення: 02.06.2025).

19. Guy Shani, David Heckerman, Ronen I. Brafman. An MDP-Based Recommender System. Journal of Machine Learning Research. Vol 6 (2005). Pp. 1265-1295.

20. Castells, P., Hurley, N.J., Vargas, S. (2015). Novelty and Diversity in Recommender Systems. In: Ricci, F., Rokach, L., Shapira, B. (eds) Recommender Systems Handbook. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7637-6_26

21. A. Chaudhari, A. A. Hitham Seddig, A. Sarlan and R. Raut, "A Hybrid Recommendation System: A Review," in IEEE Access, vol. 12, pp. 157107-157126, 2024.

22. Aggarwal C. C. Outlier Analysis. 2nd ed. New York: Springer, 2017. 466 p. DOI: https://doi.org/10.1007/978-3-319-47578-3

23. Barnett V., Lewis T. Outliers in Statistical Data. 3rd ed. Chichester: John Wiley & Sons, 1994. 584 p. DOI: https://doi.org/10.1002/bimj.4710370219

24. Chandola, Varun & Banerjee, Arindam & Kumar, Vipin. (2009). Anomaly Detection: A Survey. ACM Comput. Surv. 41. DOI: https://doi.org/10.1145/1541880.1541882

25. Hodge V., Austin J. A survey of outlier detection methodologies. Artificial Intelligence Review. 2004. Vol. 22, No. 2. P. 85–126. DOI: https://doi.org/10.1007/s10462-004-4304-y

26. Pimentel M. A. F., Clifton D. A., Clifton L., Tarassenko L. A review of novelty detection. Signal Processing. 2014. Vol. 99. P. 215–249. DOI: https://doi.org/10.1016/j.sigpro.2013.12.026

27. Goldstein M., Uchida S. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PLoS ONE. 2016. Vol. 11, No. 4. Art. 173. DOI: https://doi.org/10.1371/journal.pone.0152173

28. Hochreiter S., Schmidhuber J. Long short-term memory. Neural Computation. 1997. Vol. 9, No. 8. P. 135–160. DOI: https://doi.org/10.1162/neco.1997.9.8.1735

29. Pang G., Shen C., Cao L., Van Den Hengel A. Deep learning for anomaly detection: A review. ACM Computing Surveys. 2021. Vol. 54, No. 2. P. 381–387. DOI: https://doi.org/10.1145/3439950

30. Smola A. Introduction to machine learning / A.Smola, S.Vishwanathan [Електронний ресурс]. – Режим доступу : https://alex.smola.org/drafts/thebook.pdf.

31. Гринько, О. М. Методи системного аналізу та їх застосування у менеджменті. – Харків: Основа, 2020. 280 с.

32. Введення в машинне навчання [Електронний ресурс]. – Режим доступу // https://habr.com/ru/post/448892/

33. What is content-based filtering? [Електронний ресурс]. – Режим доступу:https://www.ibm.com/think/topics/content-based-filtering (дата звернення: 22.11.2025)

34. What is a hybrid recommender system? [Електронний ресурс]. – Режим доступу: https://zilliz.com/ai-faq/what-is-a-hybrid-recommender-system (дата звернення: 22.11.2025)

35. MacDonald M. Creating a Website: The Missing Manual / Matthew MacDonald. – 1005 Gravenstein Highway North, Sebastopol,: O’Reilly Media, Inc., 2011.

36. Tony Gaddis. Starting Out with Python. Pearson, 2018. 744 с

37. The Python Standard Library. Python. URL: https://docs.python.org/3/library/index.html (дата звернення: 07.11.2025)

38. Miguel Grinberg. Flask Web Development: Developing Web Applications with Python. O'Reilly Media, 2018. 258 с.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«Інтелектуальна рекомендаційна система для персоналізації підбору спортивного спорядження»

Виконав: студент групи ПДМ-63 Даніла РЕКУЦ

Керівник: канд. техн. наук, доцент Наталія ТРІНТІНА

Київ - 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: оптимізація підходу для підбору спортивного спорядження за допомогою інтелектуальної рекомендаційної системи з урахуванням індивідуальних характеристик, параметрів та вподобань користувача з метою формування персоналізованих рекомендацій у браузерному середовищі.

Об'єкт дослідження: процес індивідуального підбору спортивного спорядження на основі індивідуальних даних користувача та його спортивних вподобань.

Предмет дослідження: методи та алгоритми побудови гібридної рекомендаційної системи на основі контентної фільтрації для персоналізованого підбору спортивного спорядження з урахуванням профілю користувача, рівня підготовки, цілей тренувань і поведінкових патернів.

3

ПОРІВНЯННЯ МЕТОДІВ

МЕТОД РЕОМЕНДАЦІЇ	ПРИНЦИП	ПЕРЕВАГИ	ОБМЕЖЕННЯ
Content-Based Filtering	Аналіз характеристик товарів	Не потребує даних інших користувачів Пояснюваність рекомендацій	Обмежена різноманітність Складність виявлення неочевидних зв'язків Потребує детальні метадані товарів
Collaborative Filtering	Аналіз поведінки схожих користувачів	Виявляє неочевидні патерни Не залежить від контенту товарів	Cold start для нових користувачів/товарів Проблема розрідженості даних Низька точність при малій кількості оцінок
Popularity-Based	Топ товарів за рейтингом/переглядами	Простота реалізації Працює без персоналізації	Відсутність персоналізації Неефективність для досвідчених користувачів Ігнорування індивідуальних потреб

9

МАТЕМАТИЧНА МОДЕЛЬ ПРОФІЛЮ КОРИСТУВАЧА ТА ТОВАРУ

ВЕКТОР ПРОФІЛЮ КОРИСТУВАЧ

$$U = (S, L, B, G, F, C, P)$$

де S - множина видів спорту,
L - рівень підготовки,
B - бюджетні преференції,
G - множина цілей тренувань,
F - частота тренувань,
C - участь у змаганнях,
P - фізичні параметри.

ВЕКТОР ПРОФІЛЮ ТОВАРУ

$$P = (s, l, b, g, r, t, c)$$

де s — вид спорту,
l — рівень підготовки,
b — ціновий сегмент,
g — множина цілей, для яких призначений товар,
r — рейтинг,
t — тип товару,
c — категорія.

12

АЛГОРИТМ РЕКОМЕНДАЦІЙ, ПОБУДОВАНИЙ НА МЕТОДІ МАШИННОГО НАВЧАННЯ

$$R^{U_i} = \mu + b_u + b_i + q_i{}^T * p_u$$

μ – глобальне середнє значення рейтингів

b_u – хильність ставити вищі/нижчі оцінки

b_i – загальна популярність

p_u – латентний вектор користувача

q_i – латентний вектор товару

13

ГІБРІДНА МОДЕЛЬ РЕКОМЕНДАЦІЙ

Кількість оцінок	Content-Based	Collaborative	Machine Learning
< 5 (новий користувач)	**70%**	10%	20%
5-15 (середній)	**40%**	**30%**	**30%**
> 15 (досвідчений)	20%	30%	**50%**

17

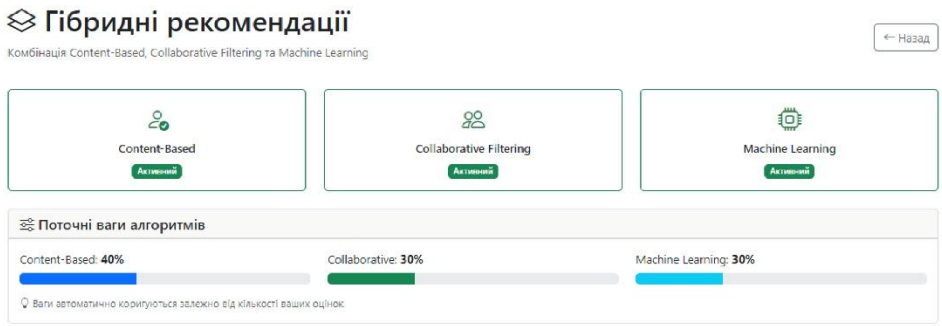
НАВЧАННЯ МОДЕЛІ РЕКОМЕНДАЦІЙ

```
✓ Завантажено оцінок: 178
✓ Користувачів: 11
✓ Товарів: 104
✓ Середня оцінка: 3.84

Розділення даних...
  Training: 142 оцінок
  Test: 36 оцінок

Параметри моделі SVD:
  Latent factors: 50
  Epochs: 20
  Learning rate: 0.005
  Regularization: 0.02
```

```
✓ Навчання завершено!

Тестування...
  Test RMSE: 1.1079
  Test MAE:  0.8092

Аналіз помилок:
  Середня: 0.8092
  Максимальна: 2.9201
  Мінімальна: 0.0369

Модель збережено: models/svd_model.pkl
Статистика: models/svd_model_stats.pkl
```

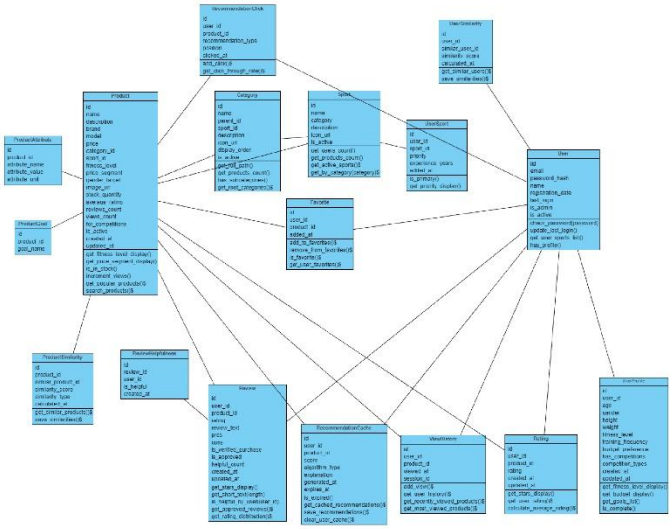
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.0895	1.2494	0.7413	0.9696	1.0433	1.0186	0.1663
MAE (testset)	0.9190	1.0269	0.5163	0.7731	0.8328	0.8136	0.1714
Fit time	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test time	0.00	0.00	0.00	0.00	0.00	0.00	0.00

```
RMSE: 1.1079
MAE: 0.8092
```

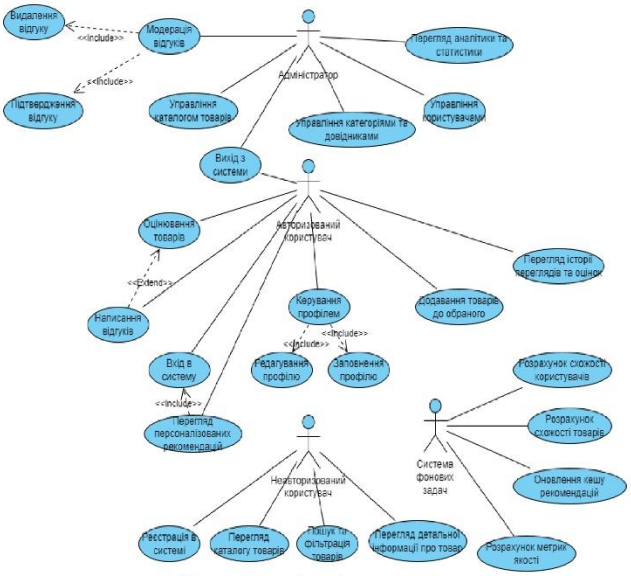


5

ДІАГРАМИ

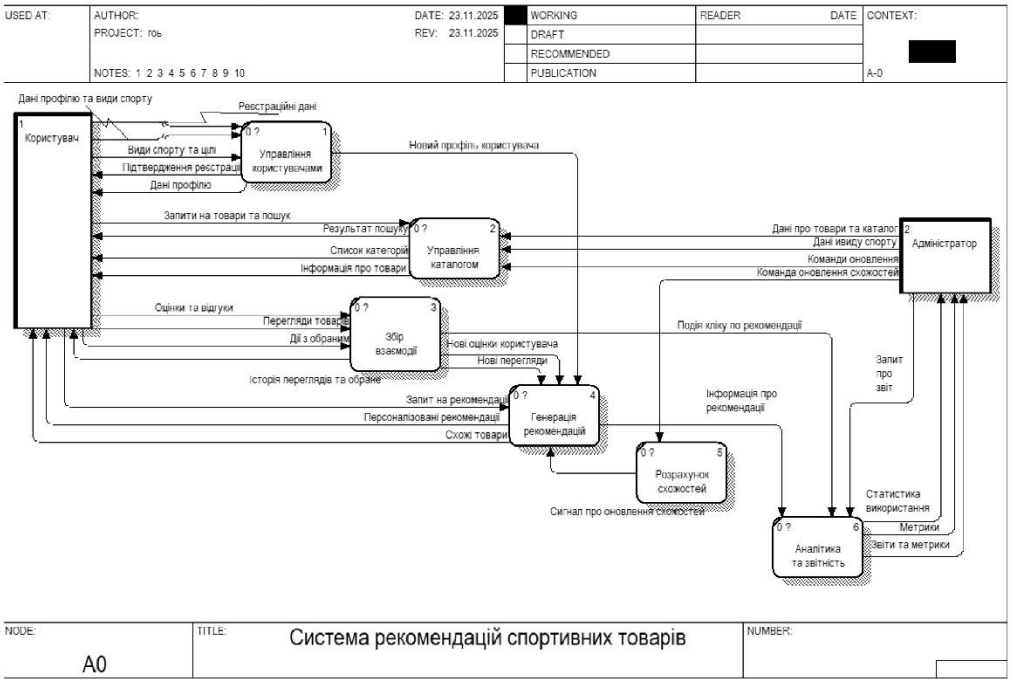


Діаграма класів

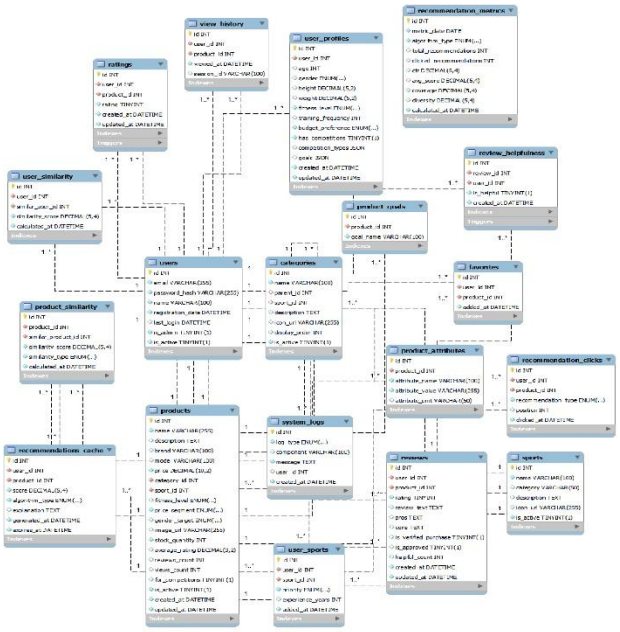


Діаграма варіантів використання

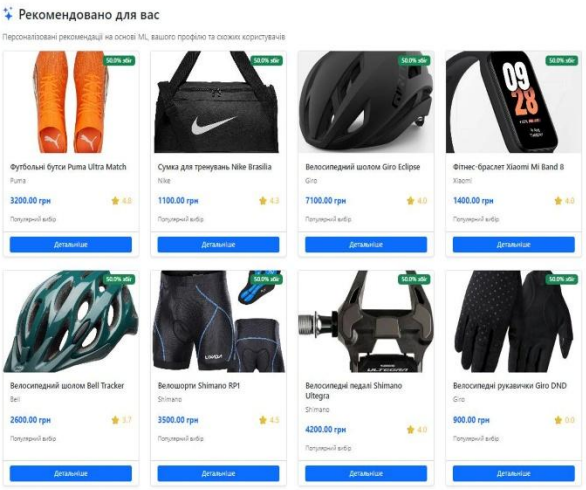
ДІАГРАМИ ПОТОКІВ ДАНИХ



МОДЕЛЬ БАЗИ ДАНИХ

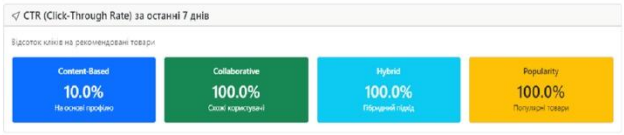


ЕКРАННІ ФОРМИ



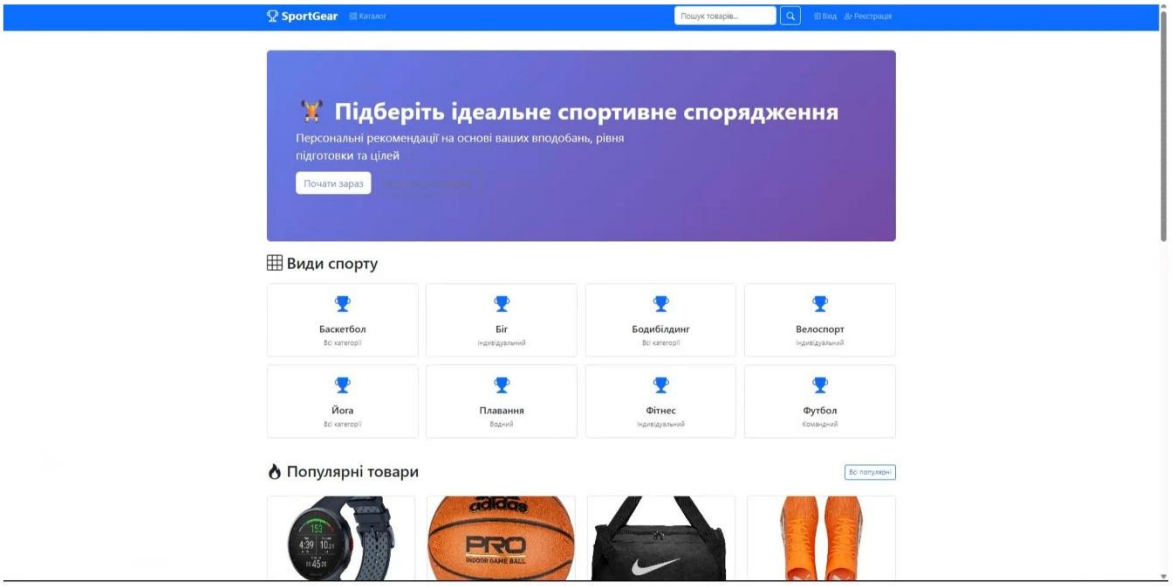
Сторінка персональні рекомендацій

Сторінка порівняння алгоритмів



Сторінка аналітики системи

ДЕМОНСТРАЦІЯ РОБОТИ ЗАСТОСУНКУ



20

ВИСНОВКИ

1. Проведено детальний аналіз традиційних підходів: Content-Based, Collaborative Filtering та Popularity-Based. Виявлено їх ключові недоліки для спортивного домену: відсутність колективного досвіду у контентних систем, проблема cold start у колаборативних та ігнорування індивідуальних параметрів у методах популярності. Доведено необхідність побудови гібридної адаптивної моделі, яка поєднує сильні сторони різних підходів та враховує специфіку спортивних товарів.

2. Створено математичну модель, що включає векторні представлення користувача та товарів, побудовані на основі критеріїв спорту, рівня підготовки, бюджету та цілей тренування. Реалізовано функції обчислення схожості (Pearson, Jaccard), які дозволили формувати релевантні рекомендації. Це забезпечило можливість адаптивної персоналізації.

3. Спроєктовано багатошарову архітектуру на основі математичної моделі рекомендацій, із виокремленими модулями обробки даних, рекомендаційного ядра та веб-інтерфейсу. Використано Flask, SQLAlchemy, MySQL, кешування та міграції БД, що забезпечило масштабованість, модульність та простоту подальшого розширення функціональності.

4. Проведено порівняльний аналіз із наявними рекомендаційними системами, що підтвердив ефективність розробленої адаптивної гібридної моделі та підвищену швидкодію завдяки попередньому розрахунку матриць схожості.

5. Створена інтелектуальна система рекомендує релевантні рекомендація для кожного користувача

21

ПУБЛІКАЦІЇ ТА АПРОБАЦІЇ РОБОТИ

Тези доповідей:

1. Рекуц Д.Е., Трінтіна Н.А. Розробка інтелектуальної системи рекомендацій для підбору спортивного спорядження. V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник матеріалів. К.: ДУІКТ, 2025. С.91.

2. Рекуц Д.Е., Трінтіна Н.А. Застосування машинного навчання у створенні персоналізованих рекомендацій спортивного інвентарю. V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник матеріалів. К.: ДУІКТ, 2025. С.130.

ДОДАТОК Б. ЛІСТІНГ ПРОГРАМНОГО КОДУ

```
from app import db

from datetime import datetime


class ViewHistory(db.Model):
    __tablename__ = 'view_history'

    id        =        db.Column(db.Integer,
primary_key=True)
    user_id      =      db.Column(db.Integer,
db.ForeignKey('users.id'), nullable=False)
    product_id    =    db.Column(db.Integer,
db.ForeignKey('products.id'),
nullable=False)
    viewed_at    =    db.Column(db.DateTime,
nullable=False,      default=datetime.utcnow,
index=True)
    session_id  =  db.Column(db.String(100),
comment='ID      сесії      для      анонімних
користувачів')

    def __repr__(self):
        return                f'<ViewHistory
user_id={self.user_id}
product_id={self.product_id}>'

    @staticmethod
    def     add_view(user_id,     product_id,
session_id=None):
        """Додати перегляд товару"""
        view = ViewHistory(
            user_id=user_id,
            product_id=product_id,
            session_id=session_id
        )
        db.session.add(view)
        db.session.commit()
        return view

    @staticmethod
    def get_user_history(user_id, limit=20):
        """Отримати    історію    переглядів
користувача"""
        return
ViewHistory.query.filter_by(user_id=user_id)
\

.order_by(ViewHistory.viewed_at.desc()) \
            .limit(limit).all()

    @staticmethod
    def
get_recently_viewed_products(user_id,
limit=10):
        """Отримати    нещодавно    переглянуті
товари (без дублікатів)"""
        from    app.models.product    import
Product

        subquery = db.session.query(
            ViewHistory.product_id,

db.func.max(ViewHistory.viewed_at).label('la
st_viewed')
        ).filter_by(user_id=user_id) \
.group_by(ViewHistory.product_id) \
            .subquery()

        products = db.session.query(Product)
\
            .join(subquery,   Product.id   ==
subquery.c.product_id) \
            .filter(Product.is_active      ==
True) \

.order_by(subquery.c.last_viewed.desc()) \
            .limit(limit).all()

        return products

    @staticmethod
    def     get_most_viewed_products(days=7,
limit=10):
        """Отримати    найбільш    переглянуті
товари за період"""
        from    app.models.product    import
Product
        from datetime import timedelta

        cutoff_date  =  datetime.utcnow()  -
timedelta(days=days)

        products = db.session.query(
            Product,

db.func.count(ViewHistory.id).label('view_co
unt')
        ).join(ViewHistory,   Product.id   ==
ViewHistory.product_id) \
            .filter(
            Product.is_active == True,
            ViewHistory.viewed_at          >=
cutoff_date
        ).group_by(Product.id) \
            .order_by(db.text('view_count
DESC')) \
            .limit(limit).all()

        return [product for product, count in
products]


class Favorite(db.Model):
    """Модель обраних товарів"""
    __tablename__ = 'favorites'

    id        =        db.Column(db.Integer,
primary_key=True)
    user_id      =      db.Column(db.Integer,
db.ForeignKey('users.id'), nullable=False)
    product_id    =    db.Column(db.Integer,
db.ForeignKey('products.id'),
nullable=False)
    added_at     =     db.Column(db.DateTime,
nullable=False, default=datetime.utcnow)

    __table_args__ = (
        db.UniqueConstraint('user_id',
'product_id',
name='unique_user_product_favorite'),
    )
```

```
    def __repr__(self):
        return                    f'<Favorite
user_id={self.user_id}
product_id={self.product_id}>'

    @staticmethod
    def             add_to_favorites(user_id,
product_id):
        """Додати товар до обраного"""
        existing                              =
Favorite.query.filter_by(user_id=user_id,
product_id=product_id).first()
        if existing:
            return existing

        favorite = Favorite(user_id=user_id,
product_id=product_id)
        db.session.add(favorite)
        db.session.commit()
        return favorite

    @staticmethod
    def        remove_from_favorites(user_id,
product_id):
        """Видалити товар з обраного"""
        favorite                              =
Favorite.query.filter_by(user_id=user_id,
product_id=product_id).first()
        if favorite:
            db.session.delete(favorite)
            db.session.commit()
            return True
        return False

    @staticmethod
    def is_favorite(user_id, product_id):
        """Перевірити чи товар в обраному"""
        return
Favorite.query.filter_by(user_id=user_id,
product_id=product_id).first() is not None

    @staticmethod
    def get_user_favorites(user_id):
        """Отримати    всі    обрані    товари
користувача"""
        from    app.models.product    import
Product

        return db.session.query(Product) \
            .join(Favorite,   Product.id   ==
Favorite.product_id) \
            .filter(
            Favorite.user_id == user_id,
            Product.is_active == True

).order_by(Favorite.added_at.desc()).all()

    @staticmethod
    def get_favorites_count(user_id):
        """Отримати     кількість     обраних
товарів"""
        return
Favorite.query.filter_by(user_id=user_id).co
unt()

    @staticmethod
    def
get_most_favorited_products(limit=10):
        """Отримати найбільш популярні товари
(найбільше в обраному)"""
        from    app.models.product    import
Product

        products = db.session.query(
            Product,

db.func.count(Favorite.id).label('favorite_c
ount')
        ).join(Favorite,    Product.id    ==
Favorite.product_id) \
            .filter(Product.is_active     ==
True) \
            .group_by(Product.id) \

.order_by(db.text('favorite_count DESC')) \
            .limit(limit).all()

        return [product for product, count in
products]

"""
Моделі товарів та категорій
"""
from app import db
from datetime import datetime
from sqlalchemy import func


class Category(db.Model):
    """Модель категорії товарів"""
    __tablename__ = 'categories'

    id        =        db.Column(db.Integer,
primary_key=True)
    name     =     db.Column(db.String(100),
nullable=False)
    parent_id    =    db.Column(db.Integer,
db.ForeignKey('categories.id'),
nullable=True)
    sport_id     =     db.Column(db.Integer,
db.ForeignKey('sports.id'), nullable=True)
    description = db.Column(db.Text)
    icon_url = db.Column(db.String(255))
    display_order   =   db.Column(db.Integer,
default=0)
    is_active     =     db.Column(db.Boolean,
nullable=False, default=True)

    # Зв'язки
    products   =   db.relationship('Product',
backref='category', lazy='dynamic')
    subcategories                           =
db.relationship('Category',
backref=db.backref('parent',
remote_side=[id]), lazy='dynamic')

    def __repr__(self):
        return f'<Category {self.name}>'

    def get_full_path(self):
        """Отримати повний шлях категорії
(Parent > Child)"""
        if self.parent:
            return   f"{self.parent.name}   >
{self.name}"
        return self.name

    def get_products_count(self):
```

```
        """Кількість активних товарів у категорії"""
        return
self.products.filter_by(is_active=True).count()

    def has_subcategories(self):
        """Перевірити чи є підкатегорії"""
        return self.subcategories.count() > 0

    @staticmethod
    def get_root_categories():
        """Отримати кореневі категорії"""
        return
Category.query.filter_by(parent_id=None,
is_active=True).order_by(Category.display_order).all()


class Product(db.Model):
    """Модель товару"""
    __tablename__ = 'products'

    id        =        db.Column(db.Integer,
primary_key=True)
    name      =      db.Column(db.String(255),
nullable=False)
    description = db.Column(db.Text)
    brand     =      db.Column(db.String(100),
index=True)
    model = db.Column(db.String(100))
    price   =   db.Column(db.Numeric(10,   2),
nullable=False)

    # Зв'язки з іншими таблицями
    category_id    =    db.Column(db.Integer,
db.ForeignKey('categories.id'),
nullable=False)
    sport_id     =     db.Column(db.Integer,
db.ForeignKey('sports.id'), nullable=False)

    # Характеристики
    fitness_level = db.Column(
        db.Enum('beginner',   'intermediate',
'advanced', 'professional', 'universal'),
        default='universal'
    )
    price_segment                              =
db.Column(db.Enum('economy',        'medium',
'premium'), nullable=False, default='medium')
    gender_target = db.Column(db.Enum('male',
'female', 'unisex'), default='unisex')

    # Зображення та наявність
    image_url = db.Column(db.String(255))
    stock_quantity   =   db.Column(db.Integer,
nullable=False, default=0)

    # Рейтинги та статистика
    average_rating = db.Column(db.Numeric(3,
2), default=0.00, comment='Середній рейтинг
0-5')
    reviews_count   =   db.Column(db.Integer,
default=0)
    views_count    =    db.Column(db.Integer,
default=0)

    # Додаткові параметри
    for_competitions = db.Column(db.Boolean,
default=False)
    is_active     =     db.Column(db.Boolean,
nullable=False, default=True)

    # Метадані
    created_at    =    db.Column(db.DateTime,
nullable=False, default=datetime.utcnow)
    updated_at    =    db.Column(db.DateTime,
nullable=False,     default=datetime.utcnow,
onupdate=datetime.utcnow)

    # Зв'язки
    attributes                                 =
db.relationship('ProductAttribute',
backref='product',           lazy='dynamic',
cascade='all, delete-orphan')
    goals  =  db.relationship('ProductGoal',
backref='product',           lazy='dynamic',
cascade='all, delete-orphan')
    ratings    =    db.relationship('Rating',
backref='product',           lazy='dynamic',
cascade='all, delete-orphan')
    reviews    =    db.relationship('Review',
backref='product',           lazy='dynamic',
cascade='all, delete-orphan')
    view_history                               =
db.relationship('ViewHistory',
backref='product',           lazy='dynamic',
cascade='all, delete-orphan')
    favorites  =  db.relationship('Favorite',
backref='product',           lazy='dynamic',
cascade='all, delete-orphan')

    def __repr__(self):
        return f'<Product {self.name}>'

    def get_fitness_level_display(self):
        """Отримати відображувану назву рівня підготовки"""
        levels = {
            'beginner': 'Початківець',
            'intermediate': 'Середній',
            'advanced': 'Просунутий',
            'professional': 'Професіонал',
            'universal': 'Універсальний'
        }
        return
levels.get(self.fitness_level,
self.fitness_level)

    def get_price_segment_display(self):
        """Отримати    відображувану    назву
цінового сегменту"""
        segments = {
            'economy': 'Економ',
            'medium': 'Середній',
            'premium': 'Преміум'
        }
        return
segments.get(self.price_segment,
self.price_segment)

    def get_gender_display(self):
        """Отримати    відображувану    назву
статі"""
        genders = {
            'male': 'Чоловічий',
            'female': 'Жіночий',
            'unisex': 'Унісекс'
        }
```

```
        return
genders.get(self.gender_target,
self.gender_target)

    def is_in_stock(self):
        """Перевірити наявність товару"""
        return self.stock_quantity > 0

    def get_rating_stars(self):
        """Отримати рейтинг у зірках (0-5)"""
        return
round(float(self.average_rating))

    def get_attributes_dict(self):
        """Отримати словник атрибутів"""
        return          {attr.attribute_name:
attr.attribute_value     for     attr     in
self.attributes.all()}

    def get_goals_list(self):
        """Отримати список цілей"""
        return  [goal.goal_name  for  goal  in
self.goals.all()]

    def increment_views(self):
        """Збільшити лічильник переглядів"""
        self.views_count += 1
        db.session.commit()

    @staticmethod
    def get_popular_products(limit=10):
        """Отримати популярні товари"""
        return
Product.query.filter_by(is_active=True) \

.order_by(Product.views_count.desc()) \
            .limit(limit).all()

    @staticmethod
    def      get_top_rated_products(limit=10,
min_reviews=5):
        """Отримати    товари    з    найвищим
рейтингом"""
        return Product.query.filter(
            Product.is_active == True,
            Product.reviews_count          >=
min_reviews

).order_by(Product.average_rating.desc()).li
mit(limit).all()

    @staticmethod
    def search_products(query, sport_id=None,
category_id=None):
        """Пошук товарів"""
        products                              =
Product.query.filter_by(is_active=True)

        if query:
            products = products.filter(
                db.or_(

Product.name.contains(query),

Product.description.contains(query),

Product.brand.contains(query)
                )
            )

        if sport_id:
            products                          =
products.filter_by(sport_id=sport_id)

        if category_id:
            products                          =
products.filter_by(category_id=category_id)

        return products.all()


class ProductAttribute(db.Model):
    """Модель атрибутів товару"""
    __tablename__ = 'product_attributes'

    id        =        db.Column(db.Integer,
primary_key=True)
    product_id    =    db.Column(db.Integer,
db.ForeignKey('products.id'),
nullable=False)
    attribute_name                            =
db.Column(db.String(100),    nullable=False,
comment='Назва атрибуту')
    attribute_value                           =
db.Column(db.String(255),    nullable=False,
comment='Значення атрибуту')
    attribute_unit = db.Column(db.String(50),
comment='Одиниця виміру')

    def __repr__(self):
        return           f'<ProductAttribute
{self.attribute_name}={self.attribute_value}
>'


class ProductGoal(db.Model):
    """Модель цілей, для яких підходить
товар"""
    __tablename__ = 'product_goals'

    id        =        db.Column(db.Integer,
primary_key=True)
    product_id    =    db.Column(db.Integer,
db.ForeignKey('products.id'),
nullable=False)
    goal_name   =   db.Column(db.String(100),
nullable=False, comment='Назва цілі')

    def __repr__(self):
        return                f'<ProductGoal
{self.goal_name}>'

"""
Моделі оцінок та відгуків
"""
from app import db
from datetime import datetime


class Rating(db.Model):
    """Модель оцінки товару"""
    __tablename__ = 'ratings'

    id        =        db.Column(db.Integer,
primary_key=True)
    user_id     =     db.Column(db.Integer,
db.ForeignKey('users.id'), nullable=False)
    product_id    =    db.Column(db.Integer,
db.ForeignKey('products.id'),
nullable=False)
```

```
    rating    =    db.Column(db.SmallInteger,
nullable=False)  # 1-5
    created_at    =    db.Column(db.DateTime,
nullable=False, default=datetime.utcnow)
    updated_at    =    db.Column(db.DateTime,
nullable=False,      default=datetime.utcnow,
onupdate=datetime.utcnow)

    # Унікальна комбінація користувач-товар
    __table_args__ = (
        db.UniqueConstraint('user_id',
'product_id',
name='unique_user_product_rating'),
        db.CheckConstraint('rating >= 1 AND
rating <= 5', name='check_rating_range')
    )

    def __repr__(self):
        return                          f'<Rating
user_id={self.user_id}
product_id={self.product_id}
rating={self.rating}>'

    def get_stars_display(self):
        """Отримати зірки для відображення"""
        return '★' * self.rating

    @staticmethod
    def get_user_rating(user_id, product_id):
        """Отримати оцінку користувача для
товару"""
        return
Rating.query.filter_by(user_id=user_id,
product_id=product_id).first()

    @staticmethod
    def
calculate_average_rating(product_id):
        """Розрахувати    середній    рейтинг
товару"""
        result = db.session.query(
            db.func.avg(Rating.rating),
            db.func.count(Rating.id)

).filter_by(product_id=product_id).first()

        avg_rating = result[0] if result[0]
else 0
        count = result[1] if result[1] else 0

        return  round(float(avg_rating),  2),
count


class Review(db.Model):
    """Модель відгуку про товар"""
    __tablename__ = 'reviews'

    id        =        db.Column(db.Integer,
primary_key=True)
    user_id      =      db.Column(db.Integer,
db.ForeignKey('users.id'), nullable=False)
    product_id    =    db.Column(db.Integer,
db.ForeignKey('products.id'),
nullable=False)
    rating    =    db.Column(db.SmallInteger,
nullable=False)  # 1-5
    review_text = db.Column(db.Text)
    pros         =         db.Column(db.Text,
comment='Переваги')
    cons         =         db.Column(db.Text,
comment='Недоліки')

    # Верифікація та модерація
    is_verified_purchase                    =
db.Column(db.Boolean, default=False)
    is_approved    =    db.Column(db.Boolean,
default=True, comment='Модерація')
    helpful_count   =   db.Column(db.Integer,
default=0,   comment='Скільки   користувачів
вважають корисним')

    # Метадані
    created_at    =    db.Column(db.DateTime,
nullable=False, default=datetime.utcnow)
    updated_at    =    db.Column(db.DateTime,
nullable=False,      default=datetime.utcnow,
onupdate=datetime.utcnow)

    # Зв'язки
    helpfulness_votes                       =
db.relationship('ReviewHelpfulness',
backref='review', lazy='dynamic',

cascade='all, delete-orphan')

    __table_args__ = (
        db.CheckConstraint('rating >= 1 AND
rating                 <=                5',
name='check_review_rating_range'),
    )

    def __repr__(self):
        return     f'<Review     id={self.id}
user_id={self.user_id}
product_id={self.product_id}>'

    def get_stars_display(self):
        """Отримати зірки для відображення"""
        return '★' * self.rating

    def get_short_text(self, length=100):
        """Отримати     скорочений     текст
відгуку"""
        if not self.review_text:
            return ''
        if len(self.review_text) <= length:
            return self.review_text
        return  self.review_text[:length]  +
'...'

    def is_helpful_by_user(self, user_id):
        """Перевірити   чи   користувач   вже
оцінив відгук як корисний"""
        return
ReviewHelpfulness.query.filter_by(
            review_id=self.id,
            user_id=user_id,
            is_helpful=True
        ).first() is not None

    @staticmethod
    def     get_approved_reviews(product_id,
limit=None):
        """Отримати   схвалені   відгуки   для
товару"""
        query                                   =
Review.query.filter_by(product_id=product_id
, is_approved=True) \
```

```
.order_by(Review.created_at.desc())

        if limit:
            query = query.limit(limit)

        return query.all()

    @staticmethod
    def get_rating_distribution(product_id):
        """Отримати розподіл оцінок (скільки відгуків з кожною оцінкою)"""
        distribution = {}
        for i in range(1, 6):
            count = Review.query.filter_by(product_id=product_id, rating=i, is_approved=True).count()
            distribution[i] = count
        return distribution


class ReviewHelpfulness(db.Model):
    """Модель корисності відгуку"""
    __tablename__ = 'review_helpfulness'

    id = db.Column(db.Integer, primary_key=True)
    review_id = db.Column(db.Integer, db.ForeignKey('reviews.id'), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    is_helpful = db.Column(db.Boolean, nullable=False)
    created_at = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)

    # Унікальна комбінація відгук-користувач
    __table_args__ = (
        db.UniqueConstraint('review_id', 'user_id', name='unique_review_user'),
    )

    def __repr__(self):
        return f'<ReviewHelpfulness review_id={self.review_id} user_id={self.user_id}>'

"""
Моделі системи рекомендацій
"""
from app import db
from datetime import datetime, timedelta


class RecommendationCache(db.Model):
    """Модель кешу рекомендацій"""
    __tablename__ = 'recommendations_cache'

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    product_id = db.Column(db.Integer, db.ForeignKey('products.id'), nullable=False)
    score = db.Column(db.Numeric(5, 4), nullable=False, comment='Оцінка рекомендації 0-1')
    algorithm_type = db.Column(
        db.Enum('content_based', 'collaborative', 'hybrid', 'popularity'),
        nullable=False
    )
    explanation = db.Column(db.Text, comment='Пояснення чому рекомендовано')
    generated_at = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    expires_at = db.Column(db.DateTime, nullable=False, comment='Час закінчення актуальності')

    def __repr__(self):
        return f'<RecommendationCache user_id={self.user_id} product_id={self.product_id}>'

    def is_expired(self):
        """Перевірити чи кеш застарів"""
        return datetime.utcnow() > self.expires_at

    @staticmethod
    def get_cached_recommendations(user_id, algorithm_type=None, limit=10):
        """Отримати кешовані рекомендації для користувача"""
        from app.models.product import Product

        query = db.session.query(Product, RecommendationCache) \
            .join(RecommendationCache, Product.id == RecommendationCache.product_id) \
            .filter(
            RecommendationCache.user_id == user_id,
            RecommendationCache.expires_at > datetime.utcnow(),
            Product.is_active == True
        )

        if algorithm_type:
            query = query.filter(RecommendationCache.algorithm_type == algorithm_type)

        results = query.order_by(RecommendationCache.score.desc()).limit(limit).all()

        return [(product, cache.score, cache.explanation) for product, cache in results]

    @staticmethod
    def save_recommendations(user_id, recommendations, algorithm_type, expiry_hours=24):
        """Зберегти рекомендації в кеш"""
        expires_at = datetime.utcnow() + timedelta(hours=expiry_hours)

        # Видалити старі рекомендації для цього користувача та алгоритму
        RecommendationCache.query.filter_by(
            user_id=user_id,
            algorithm_type=algorithm_type
        ).delete()
```

```
        # Додати нові рекомендації
        for product_id, score, explanation in
recommendations:
            cache = RecommendationCache(
                user_id=user_id,
                product_id=product_id,
                score=score,

algorithm_type=algorithm_type,
                explanation=explanation,
                expires_at=expires_at
            )
            db.session.add(cache)

        db.session.commit()

    @staticmethod
    def clear_user_cache(user_id):
        """Очистити кеш рекомендацій
користувача"""

RecommendationCache.query.filter_by(user_id=
user_id).delete()
        db.session.commit()

    @staticmethod
    def clear_expired_cache():
        """Очистити застарілий кеш"""
        RecommendationCache.query.filter(
            RecommendationCache.expires_at
<= datetime.utcnow()
        ).delete()
        db.session.commit()


class RecommendationClick(db.Model):
    """Модель кліків на рекомендації"""
    __tablename__ = 'recommendation_clicks'

    id        =        db.Column(db.Integer,
primary_key=True)
    user_id     =     db.Column(db.Integer,
db.ForeignKey('users.id'), nullable=False)
    product_id    =    db.Column(db.Integer,
db.ForeignKey('products.id'),
nullable=False)
    recommendation_type = db.Column(
        db.Enum('content_based',
'collaborative',   'hybrid',   'popularity',
'similar_products'),
        nullable=False
    )
    position     =     db.Column(db.Integer,
comment='Позиція в списку рекомендацій')
    clicked_at    =    db.Column(db.DateTime,
nullable=False,     default=datetime.utcnow,
index=True)

    def __repr__(self):
        return         f'<RecommendationClick
user_id={self.user_id}
product_id={self.product_id}>'

    @staticmethod
    def    add_click(user_id,    product_id,
recommendation_type, position=None):
        """Додати клік на рекомендацію"""
        click = RecommendationClick(
            user_id=user_id,
            product_id=product_id,

recommendation_type=recommendation_type,
            position=position
        )
        db.session.add(click)
        db.session.commit()
        return click

    @staticmethod
    def
get_click_through_rate(algorithm_type,
days=7):
        """Розрахувати CTR для алгоритму"""
        cutoff_date  =  datetime.utcnow()  -
timedelta(days=days)

        total_recommendations               =
RecommendationCache.query.filter(

RecommendationCache.algorithm_type         ==
algorithm_type,
            RecommendationCache.generated_at
>= cutoff_date
        ).count()

        total_clicks                        =
RecommendationClick.query.filter(

RecommendationClick.recommendation_type    ==
algorithm_type,
            RecommendationClick.clicked_at
>= cutoff_date
        ).count()

        if total_recommendations == 0:
            return 0.0

        return      round(total_clicks      /
total_recommendations, 4)


class ProductSimilarity(db.Model):
    """Модель схожості товарів"""
    __tablename__ = 'product_similarity'

    id        =        db.Column(db.Integer,
primary_key=True)
    product_id    =    db.Column(db.Integer,
db.ForeignKey('products.id'),
nullable=False)
    similar_product_id                      =
db.Column(db.Integer,
db.ForeignKey('products.id'),
nullable=False)
    similarity_score                        =
db.Column(db.Numeric(5,  4),  nullable=False,
comment='Оцінка схожості 0-1')
    similarity_type                         =
db.Column(db.Enum('content',
'collaborative', 'hybrid'), nullable=False)
    calculated_at   =  db.Column(db.DateTime,
nullable=False, default=datetime.utcnow)

    def __repr__(self):
        return           f'<ProductSimilarity
product_id={self.product_id}
similar={self.similar_product_id}>'

    @staticmethod
```

```
    def get_similar_products(product_id, limit=5, min_score=0.5):
        """Отримати схожі товари"""
        from app.models.product import Product

        similar = db.session.query(Product, ProductSimilarity.similarity_score) \
            .join(ProductSimilarity, Product.id == ProductSimilarity.similar_product_id) \
            .filter(
            ProductSimilarity.product_id == product_id,
            ProductSimilarity.similarity_score >= min_score,
            Product.is_active == True
        ).order_by(ProductSimilarity.similarity_score.desc()) \
            .limit(limit).all()

        return [(product, score) for product, score in similar]

    @staticmethod
    def save_similarities(product_id, similarities, similarity_type='content'):
        """Зберегти схожість товарів"""
        # Видалити старі записи схожості для цього товару
        ProductSimilarity.query.filter_by(
            product_id=product_id,
            similarity_type=similarity_type
        ).delete()

        # Додати нові записи
        for similar_id, score in similarities:
            similarity = ProductSimilarity(
                product_id=product_id,
                similar_product_id=similar_id,
                similarity_score=score,
                similarity_type=similarity_type
            )
            db.session.add(similarity)

        db.session.commit()


class UserSimilarity(db.Model):
    """Модель схожості користувачів"""
    __tablename__ = 'user_similarity'

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    similar_user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    similarity_score = db.Column(db.Numeric(5, 4), nullable=False, comment='Оцінка схожості 0-1')
    calculated_at = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)

    def __repr__(self):
        return f'<UserSimilarity user_id={self.user_id} similar={self.similar_user_id}>'

    @staticmethod
    def get_similar_users(user_id, limit=10, min_score=0.5):
        """Отримати схожих користувачів"""
        from app.models.user import User

        similar = db.session.query(User, UserSimilarity.similarity_score) \
            .join(UserSimilarity, User.id == UserSimilarity.similar_user_id) \
            .filter(
            UserSimilarity.user_id == user_id,
            UserSimilarity.similarity_score >= min_score,
            User.is_active == True
        ).order_by(UserSimilarity.similarity_score.desc()) \
            .limit(limit).all()

        return [(user, score) for user, score in similar]

    @staticmethod
    def save_similarities(user_id, similarities):
        """Зберегти схожість користувачів"""
        # Видалити старі записи
        UserSimilarity.query.filter_by(user_id=user_id).delete()

        # Додати нові записи
        for similar_id, score in similarities:
            similarity = UserSimilarity(
                user_id=user_id,
                similar_user_id=similar_id,
                similarity_score=score
            )
            db.session.add(similarity)

        db.session.commit()
"""
Моделі видів спорту
"""
from app import db
from datetime import datetime


class Sport(db.Model):
    """Модель виду спорту"""
    __tablename__ = 'sports'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), unique=True, nullable=False)
    category = db.Column(db.String(50), comment='Категорія: індивідуальний, командний тощо')
    description = db.Column(db.Text)
    icon_url = db.Column(db.String(255))
    is_active = db.Column(db.Boolean, nullable=False, default=True)
```

```
    # Зв'язки
    user_sports = db.relationship('UserSport',
backref='sport', lazy='dynamic')
    products = db.relationship('Product',
backref='sport', lazy='dynamic')
    categories = db.relationship('Category',
backref='sport', lazy='dynamic')

    def __repr__(self):
        return f'<Sport {self.name}>'

    def get_users_count(self):
        """Кількість користувачів, які
займаються цим видом спорту"""
        return self.user_sports.count()

    def get_products_count(self):
        """Кількість товарів для цього виду
спорту"""
        return
self.products.filter_by(is_active=True).coun
t()

    @staticmethod
    def get_active_sports():
        """Отримати всі активні види
спорту"""
        return
Sport.query.filter_by(is_active=True).order_
by(Sport.name).all()

    @staticmethod
    def get_by_category(category):
        """Отримати види спорту за
категорією"""
        return
Sport.query.filter_by(category=category,
is_active=True).all()


class UserSport(db.Model):
    """Модель зв'язку користувача та виду
спорту"""
    __tablename__ = 'user_sports'

    id = db.Column(db.Integer,
primary_key=True)
    user_id = db.Column(db.Integer,
db.ForeignKey('users.id'), nullable=False)
    sport_id = db.Column(db.Integer,
db.ForeignKey('sports.id'), nullable=False)
    priority = db.Column(db.Enum('primary',
'secondary'), nullable=False,
default='primary')
    experience_years = db.Column(db.Integer,
default=0)
    added_at = db.Column(db.DateTime,
nullable=False, default=datetime.utcnow)

    # Унікальна комбінація користувач-спорт
    __table_args__ = (
        db.UniqueConstraint('user_id',
'sport_id', name='unique_user_sport'),
    )

    def __repr__(self):
        return f'<UserSport
user_id={self.user_id}
sport_id={self.sport_id}>'

    def is_primary(self):
        """Перевірити чи це основний вид
спорту"""
        return self.priority == 'primary'

    def get_priority_display(self):
        """Отримати відображувану назву
пріоритету"""
        priorities = {
            'primary': 'Основний',
            'secondary': 'Додатковий'
        }
        return priorities.get(self.priority,
self.priority)
"""
Моделі користувачів та профілів
"""
from app import db, login_manager
from flask_login import UserMixin
from werkzeug.security import
generate_password_hash, check_password_hash
from datetime import datetime


class User(UserMixin, db.Model):
    """Модель користувача"""
    __tablename__ = 'users'

    id = db.Column(db.Integer,
primary_key=True)
    email = db.Column(db.String(255),
unique=True, nullable=False, index=True)
    password_hash = db.Column(db.String(255),
nullable=False)
    name = db.Column(db.String(100),
nullable=False)
    registration_date =
db.Column(db.DateTime, nullable=False,
default=datetime.utcnow)
    last_login = db.Column(db.DateTime)
    is_admin = db.Column(db.Boolean,
nullable=False, default=False)
    is_active = db.Column(db.Boolean,
nullable=False, default=True)

    # Зв'язки
    profile = db.relationship('UserProfile',
backref='user', uselist=False, cascade='all,
delete-orphan')
    sports = db.relationship('UserSport',
backref='user', lazy='dynamic', cascade='all,
delete-orphan')
    ratings = db.relationship('Rating',
backref='user', lazy='dynamic', cascade='all,
delete-orphan')
    reviews = db.relationship('Review',
backref='user', lazy='dynamic', cascade='all,
delete-orphan')
    view_history =
db.relationship('ViewHistory',
backref='user', lazy='dynamic', cascade='all,
delete-orphan')
    favorites = db.relationship('Favorite',
backref='user', lazy='dynamic', cascade='all,
delete-orphan')
```

```
    def __repr__(self):
        return f'<User {self.email}>'

    def set_password(self, password):
        """Встановити пароль (хешування)"""
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        """Перевірити пароль"""
        return check_password_hash(self.password_hash, password)

    def update_last_login(self):
        """Оновити час останнього входу"""
        self.last_login = datetime.utcnow()
        db.session.commit()

    def get_user_sports_list(self):
        """Отримати список видів спорту користувача"""
        return [us.sport for us in self.sports.all()]

    def has_profile(self):
        """Перевірити чи є профіль"""
        return self.profile is not None


class UserProfile(db.Model):
    """Модель профілю користувача"""
    __tablename__ = 'user_profiles'

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), unique=True, nullable=False)

    # Фізичні параметри
    age = db.Column(db.Integer)
    gender = db.Column(db.Enum('male', 'female', 'other'), default=None)
    height = db.Column(db.Numeric(5, 2), comment='Зріст у см')
    weight = db.Column(db.Numeric(5, 2), comment='Вага у кг')

    # Спортивні параметри
    fitness_level = db.Column(
        db.Enum('beginner', 'intermediate', 'advanced', 'professional'),
        nullable=False,
        default='beginner'
    )
    training_frequency = db.Column(db.Integer, comment='Тренувань на тиждень')
    budget_preference = db.Column(
        db.Enum('economy', 'medium', 'premium'),
        nullable=False,
        default='medium'
    )

    # Змагання
    has_competitions = db.Column(db.Boolean, nullable=False, default=False)
    competition_types = db.Column(db.JSON, comment='Масив типів змагань')

    # Цілі
    goals = db.Column(db.JSON, comment='Масив цілей тренувань')

    # Метадані
    created_at = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    updated_at = db.Column(db.DateTime, nullable=False, default=datetime.utcnow, onupdate=datetime.utcnow)

    def __repr__(self):
        return f'<UserProfile user_id={self.user_id}>'

    def get_fitness_level_display(self):
        """Отримати відображувану назву рівня підготовки"""
        levels = {
            'beginner': 'Початківець',
            'intermediate': 'Середній',
            'advanced': 'Просунутий',
            'professional': 'Професіонал'
        }
        return levels.get(self.fitness_level, self.fitness_level)

    def get_budget_display(self):
        """Отримати відображувану назву бюджету"""
        budgets = {
            'economy': 'Економ',
            'medium': 'Середній',
            'premium': 'Преміум'
        }
        return budgets.get(self.budget_preference, self.budget_preference)

    def get_goals_list(self):
        """Отримати список цілей"""
        return self.goals if self.goals else []

    def get_competition_types_list(self):
        """Отримати список типів змагань"""
        return self.competition_types if self.competition_types else []

    def is_complete(self):
        """Перевірити чи профіль заповнений"""
        required_fields = [
            self.fitness_level,
            self.training_frequency,
            self.budget_preference
        ]
        return all(required_fields) and len(self.user.sports.all()) > 0


@login_manager.user_loader
def load_user(user_id):
    """Завантажити користувача для Flask-Login"""
    return User.query.get(int(user_id)
```