Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка інтелектуального застосунку для тренерів і відвідувачів тренажерної зали з автоматичною генерацією тренувальних програм на основі фізіологічних параметрів користувача
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-4
Семен Животов
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник доц. Олена Волощук
(посада, власне ім'я, прізвище)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

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

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Животову Семену Костянтиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтелектуального застосунку для тренерів і відвідувачів тренажерної зали з автоматичною генерацією тренувальних програм на основі фізіологічних параметрів користувача
затверджена наказом університету від 19 травня 20 25 р. № 378Ст
2. Термін подання студентом роботи до екзаменаційної комісії 25 червня 20 25 р.
3. Вихідні дані до роботи Відкриті датасети тренувальних програм та фізичних вправ; набори антропометричних показників користувачів для формування персональних рекомендацій; міжнародні нормативні документи та стандарти з фітнес-тренувань (ISO 20957, ДСТУ ISO 7250-1, ACSM Guidelines); наукові статті та відкриті дослідження у галузі персоналізації фізичних навантажень, спортивної медицини та застосування методів штучного інтелекту для рекомендацій; інструменти та бібліотеки машинного навчання для побудови рекомендаторних систем

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Теоретичне дослідження
3) Практична реалізація

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Строк / терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання на кваліфікаційну роботу | 19.05.2025 | виконано |
| 2 | Аналіз предметної галузі та постановка задачі | 20.05.2025 | виконано |
| 3 | Теоретичне дослідження | 20.05.2025 | виконано |
| 4 | Практична реалізація | 23.05.2025 | виконано |
| 5 | Написання пояснювальної записки | 25.05.2025 | виконано |
| 6 | Перевірка на академічний плагіат | 26.05.2025 | виконано |
| 7 | Нормоконтроль | 06.06.2025 | виконано |
| 8 | Підготовка презентації та доповіді | 10.06.2025 | виконано |
| 9 | Попередній захист | 13.06.2025 | виконано |
| 10 | Рецензування | 23.06.2025 | виконано |
| 11 | Захист перед ЕК | 25.06.2025 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олена Волощук
(підпис) (посада, власне ім’я, прізвище)

# РЕФЕРАТ

Пояснювальна записка: 60 с., 6 рис., 2 дод., 17 джерел.

ГЕНЕРАЦІЯ ПРОГРАМ, СИСТЕМА РЕКОМЕНДАЦІЙ, СПОРТИВНА АНАЛІТИКА, ТРЕНАЖЕРНА ЗАЛА, ТРЕНУВАЛЬНА ПРОГРАМА, ФІЗІОЛОГІЧНІ ПАРАМЕТРИ, ШТУЧНИЙ ІНТЕЛЕКТ, JAVA, POSTGRESQL, SPRING.

Об'єкт дослідження – процес генерації персоналізованих тренувальних програм для користувачів тренажерної зали з використанням сучасних інформаційних технологій та штучного інтелекту.

Предмет дослідження – інтелектуальна система автоматичної генерації тренувальних програм на основі фізіологічних параметрів користувача.

Мета роботи – розробка інтелектуального застосунку для тренерів і відвідувачів тренажерної зали з автоматичною генерацією тренувальних програм на основі фізіологічних параметрів користувача.

Методи дослідження – аналіз літературних джерел, вивчення аналогічних систем, теоретичне дослідження принципів роботи рекомендаційних систем та застосування штучного інтелекту, архітектурне моделювання та практична апробація.

У результаті виконання звіту з практики проаналізовано сучасні підходи до автоматизації створення тренувальних програм, обґрунтовано вибір методів штучного інтелекту та розроблено основні компоненти серверної частини майбутнього застосунку.

# ABSTRACT

Bachelor’s thesis contains: 60 pp., 6 fig., 2 ann., 17 references.

ARTIFICIAL INTELLIGENCE, GYM, JAVA, PHYSIOLOGICAL PARAMETERS, POSTGRESQL, PROGRAM GENERATION, RECOMMENDER SYSTEM, SPORTS ANALYTICS, SPRING, TRAINING PROGRAM.

The object of the research is the process of generating personalized training programs for gym users using modern information technologies and artificial intelligence.

The subject of the research is the intelligent system for automatic generation of training programs based on the physiological parameters of the user.

The purpose of the work is to develop an intelligent application for trainers and gym visitors with automatic generation of training programs based on the physiological parameters of the user.

Research methods – analysis of literary sources, study of analogous systems, theoretical research of recommender system principles and artificial intelligence applications, architectural modeling, and practical testing.

As a result of the practice report, modern approaches to the automation of training program creation were analyzed, the choice of artificial intelligence methods was substantiated, and the main components of the backend of the future application were developed.

**ЗМІСТ**

# ВСТУП

У сучасному світі цифрові інформаційні технології дедалі глибше інтегруються у різні аспекти життя, бізнесу та спорту. Особливої ваги набуває використання інноваційних технологій штучного інтелекту (ШІ), які дозволяють автоматизувати та персоналізувати процеси у реальному часі. Одним із перспективних напрямків застосування ШІ є організація та супровід індивідуальних фізичних тренувань, де важливо враховувати унікальні фізіологічні особливості, спортивний досвід, поточний стан здоров’я та цілі користувача.

Традиційний підхід у створенні тренувальних програм передбачає значні витрати праці та часу як від тренерів, так і самих спортсменів, а індивідуальні потреби часто залишаються поза увагою через обмеження людського фактора. Зростання числа користувачів фітнес-програм, розвиток цифрових платформ і онлайн-сервісів зумовлює попит на автоматизовані та адаптивні рішення, які здатні формувати персоналізований тренувальний процес із залученням сучасних рекомендаційних систем та алгоритмів обробки даних.

Саме тому розробка інтелектуального програмного забезпечення для тренажерної зали, що автоматично підбирає та генерує тренувальні плани на основі фізіологічних параметрів користувача і прогресу занять, є актуальною науково-технічною задачею сучасності. В роботі особливий фокус зроблено на аналізі можливостей застосування штучного інтелекту, сучасних підходів до рекомендацій, а також їх програмній реалізації на базі новітнього стеку технологій.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Актуальність інтелектуальних систем у фітнес-індустрії

У сучасному суспільстві питання здоров’я та активного способу життя стають пріоритетними як для державної політики різних країн, так і для громадян, які дедалі частіше прагнуть зробити здоровий спосіб життя своєю щоденною звичкою. В останні роки значно зросла кількість людей, що відвідують тренажерні зали та спортивні клуби, використовуючи різноманітні онлайн-сервіси й мобільні додатки для вибору тренувальних програм, контролю прогресу й досягнення індивідуальних цілей у спорті. При цьому користувачі сучасних фітнес-платформ очікують отримати не лише стандартні набори рекомендацій, а й персоналізовані підходи, які спираються на конкретні фізіологічні та особистісні параметри: вік, стать, антропометричні характеристики, наявні захворювання або травми, рівень фізичної підготовки, минулий досвід занять та персональні цілі, такі як схуднення, набір м’язової маси, підтримка фізичного стану чи відновлення після травм [4].

Індивідуалізовані тренувальні програми мають низку суттєвих переваг. Так, дослідження продемонстрували, що персоналізація фізичних навантажень дозволяє збільшити ефективність та мотивацію до регулярних тренувань, знизити ймовірність виникнення травм шляхом врахування стану здоров’я і фізичної готовності користувача та покращити кінцеві результати у значно коротші терміни [2]. У той же час, традиційний підхід, що передбачає ручне складання персонального плану тренером-експертом, має суттєві недоліки в контексті масштабування. Зокрема, такий спосіб створення тренувальних програм обмежений за швидкістю, якістю та кількістю одночасно сформованих індивідуальних планів, схильний до суб'єктивних помилок і недоліків, що ведуть до втрати важливих

індивідуальних характеристик користувача, які могли б суттєво вплинути на ефективність занять [1].

Прикладні дослідження також вказують на те, що ефективні персоналізовані тренувальні плани потребують не лише статичної інформації, але й врахування динамічних даних про прогрес користувача, його реакцію організму на вправи, історію навантажень та регулярність занять [2], [3]. Практичне впровадження таких рішень є складним завданням, яке неможливо реалізувати вручну для великої кількості користувачів без суттєвої втрати точності персоналізації. Виникає необхідність у методах автоматизованої персоналізації: аналізі часових послідовностей тренувальних занять, адаптивному моніторингу реакції людини, побудові моделей передбачення ефекту від конкретних вправ [2], [3], [4].

Значні перспективи відкриваються завдяки розвитку технологій штучного інтелекту (ШІ), машинного навчання (Machine Learning), глибокого навчання (Deep Learning) та аналізу великих даних, які сьогодні дозволяють реалізувати високоефективні системи для побудови автоматизованих персональних рекомендацій [1], [4]. Такі системи можуть ефективно зчитувати, аналізувати і обробляти широкі набори даних від користувачів, представлені як у вигляді структурованих, так і неструктурованих даних – інформації з датчиків, фітнес-трекерів, медичних карток, описів вправ та тренувань, тексту коментарів і персональних відгуків [1], [3], [5].

Успішні кейси провідних глобальних брендів у сфері спорту і фітнесу, таких як Adidas (додаток Adidas Running та Runtastic, що використовує AI для персоналізації тренувань [6]), а також Fitbod – популярна платформа, що пропонує персоналізовані тренувальні плани за допомогою ШІ [7], демонструють ефективність та комерційну перспективність такого підходу. Технології ШІ в таких випадках забезпечують фізичну активність користувачів більш підходящими вправами, підвищуючи мотивацію,

ефективність та постійність тренувань [7]. Крім того, такі інтелектуальні рішення дозволяють знизити витрати часу та ресурсів на підготовку тренувальних програм, оптимізують робочі процеси фітнес-клубів та зменшують імовірність помилок чи некоректних рекомендацій, що можуть негативно вплинути на здоров’я користувачів [1], [4].

Таким чином, необхідність впровадження інтелектуальних рекомендаторних систем для автоматизації процесів персоналізації тренувальних програм актуальна з точки зору як потреб суспільства та користувачів, так і бізнес-інтересів провідних представників спортивної та фітнес-галузі. Реалізація подібних систем здатна значно покращити якість і результативність занять фізичними вправами, сприяти популяризації активного способу життя серед широкого кола людей та відкрити нові можливості для розвитку цифрових проєктів у сфері здоров'я й фітнесу.

## 1.2 Технологічні тренди та сучасний стан автоматизації

Останні роки характеризуються суттєвим прогресом у автоматизації процесів в галузі фітнес-послуг і спорту, яка стала можливою завдяки інтеграції сучасних інформаційних технологій, цифрових платформ та інноваційних підходів до взаємодії з користувачем. Фітнес-індустрія активно впроваджує цифрові рішення, що дозволяють не тільки автоматизувати рутинні процеси, а й якісно підвищити рівень персоналізації тренувальних рекомендацій та результативність фізичних тренувань [2], [4].

Архітектурною та технологічною основою сучасних цифрових систем у сфері фітнесу і здоров’я стають масштабовані хмарні платформи (cloud-based solutions), серверні веб-додатки (web-based applications), RESTful API, потужні і ефективні системи керування базами даних (наприклад, PostgreSQL, MongoDB, MySQL). Такі рішення дають змогу забезпечити централізовану обробку, безпечне зберігання великих обсягів особистих даних, а також оперативну взаємодію з кінцевими користувачами. Також

вони створюють необхідні умови для постійного збору й аналізу інформації у режимі реального часу, формуючи основу для управлінських рішень та кращої персоналізації послуг [1], [4].

Особливо затребуваними стають web-сервіси та мобільні застосунки, які забезпечують зручний доступ для користувачів до оцінки власної фізичної активності, прогресу та самопочуття. Завдяки простим та інтуїтивним інтерфейсам, користувач може легко надавати необхідні для рекомендацій дані, отримувати зворотний зв’язок від платформи, а також переглядати візуалізовані результати своїх тренувань й динаміку змін фізичних показників [2], [6].

Ключовими сучасними тенденціями розвитку автоматизованих фітнес-систем сьогодні виступають наступні:

– систематизація й автоматичний збір індивідуальних даних користувача – це включає збір історії тренувань, показників фізичної активності (частота тренувань, тривалість занять, використовувані вправи), результатів медичних обстежень та вимірювань біометричних параметрів (вага, пульс, тиск, калорійність споживання), а також автоматизацію анкетування для визначення персональних цілей, особливих умов користувача, наявних травм чи захворювань тощо [2], [3];

– персоналізовані модулі рекомендацій – впровадження інтелектуальних алгоритмів персоналізації на базі методів машинного навчання (Machine Learning, ML), глибокого навчання (Deep Learning) і технологій Big Data, що дозволяють максимально точно підбирати оптимальні навантаження, інтенсивність занять та їх тип залежно від індивідуальних можливостей і цілей конкретного користувача. В результаті такого підходу суттєво підвищується ефективність тренувань, зменшується вірогідність помилок і травм, прискорюється досягнення бажаних результатів [1], [2], [4];

– створення комплексних моделей для коротко- та довгострокового планування тренувань на основі передбачувальних моделей та аналізу

часових послідовностей (time-series forecasting). Сучасні IT-рішення дозволяють побудувати адаптивні системи, що враховують не тільки поточний стан користувача, але й прогнозують потенційні зміни його фізичного стану при регулярних заняттях. Це сприяє формуванню не лише тактичних, а й стратегічних підходів до планування тренувальних навантажень користувача [3], [4], [5];

– підвищення ролі аналітики та її інтеграції в інтерфейс взаємодії з користувачем. Сучасні тренувальні платформи дедалі активніше впроваджують технології інтерактивних dashboard-систем, наочних графіків та інформативного візуального представлення ефективності занять. Це допомагає не тільки краще зрозуміти власні результати, а й покращити довіру та залученість користувачів до платформи завдяки прозорості показників і більш якісному зворотному зв'язку [1], [6].

Таким чином, сучасні digital-рішення у галузі фітнесу виходять за межі звичайних персональних калькуляторів чи щоденників тренувань, трансформуючись у повноцінних, інтелектуальних і високотехнологічних помічників користувача, що взаємодіють з ним в інтерактивній формі, забезпечують високий рівень персоналізації та адаптивності, використовуючи сучасні досягнення штучного інтелекту, машинного навчання та аналізу великих даних. Подібні технологічні рішення вже зараз демонструють високу ефективність та перспективи для подальшого втілення й удосконалення на ринку фітнес-індустрії [1], [5], [6].

## 1.3 Огляд і аналіз існуючих рішень

Сьогодні на ринку існує чимало відомих фітнес-додатків і платформ: Nike Training Club, Freeletics, Adidas Training, Fitbod, Jefit, MyFitnessPal і багато інших [7]. Аналіз їх функціоналу свідчить, що переважна більшість з них функціонує за одним із трьох принципів:

Побудова типових програм за шаблоном («пакет рішень для всіх»). Обмежена варіативність, низький рівень персоналізації.

Врахування фізіологічних характеристик («ручний» підбір під час реєстрації/анкети): дані враховуються на вході, але далі програма не адаптується до змін користувача в динаміці [4].

Використання окремих ML-підходів та rule-based систем: більшість рекомендаційних рішень будується на простих алгоритмах («rule-based reasoning»), які не враховують попередні результати занять та схожість з іншими користувачами.

Зазначимо, що деякі додатки впровадили більш складні підходи персоналізації: Fitbod використовує машинне навчання для підбору вправ на основі попередніх тренувань і циклів (там зібрана значна база статичних і динамічних параметрів користувача) [7], Freeletics декларує модуль AI-тренера, який удосконалюється на основі великих наборів анонімізованих даних.

Втім, більшість систем не оперує справжньою автоматизованою персоналізацією на рівні сучасних методів штучного інтелекту:

– динамічного врахування змін у параметрах користувача не відбувається;

– пояснювання рекомендацій відсутня;

– складні рекомендаційні алгоритми впроваджені обмежено;

– взаємодії між платформами бракує (закриті екосистеми);

– немає можливості легко масштабувати персональний досвід на всі категорії користувачів [5].

Ці фактори створюють простір для вдосконалення та потребують створення нового покоління AI-орієнтованих, відкритих та адаптивних фітнес-платформ [1].

## 1.4 Постановка задачі

На підставі аналізу існуючих недоліків і нових викликів ринку, можна сформулювати основну інженерну та наукову задачу даної роботи:

– розробити програмний продукт із сучасною серверною архітектурою (Spring Boot, PostgreSQL, REST API), що забезпечуватиме автоматичне формування персоналізованих тренувальних програм на основі даних користувача;

– впровадити в систему модулі машинного навчання та штучного інтелекту, які дозволять перейти від шаблонного підходу до справжньої динамічної та гнучкої персоналізації програми;

– забезпечити архітектурну відкритість, масштабованість та можливість подальшого розвитку (модульність, можливість впровадження нових типів рекомендацій);

– досягти максимального рівня прозорості та пояснюваності рекомендацій для користувача, що сприятиме підвищенню лояльності і довіри до системи [1].

Концептуальну архітектуру майбутньої системи та основні дані її потоків відображено на рисунку 1.1.

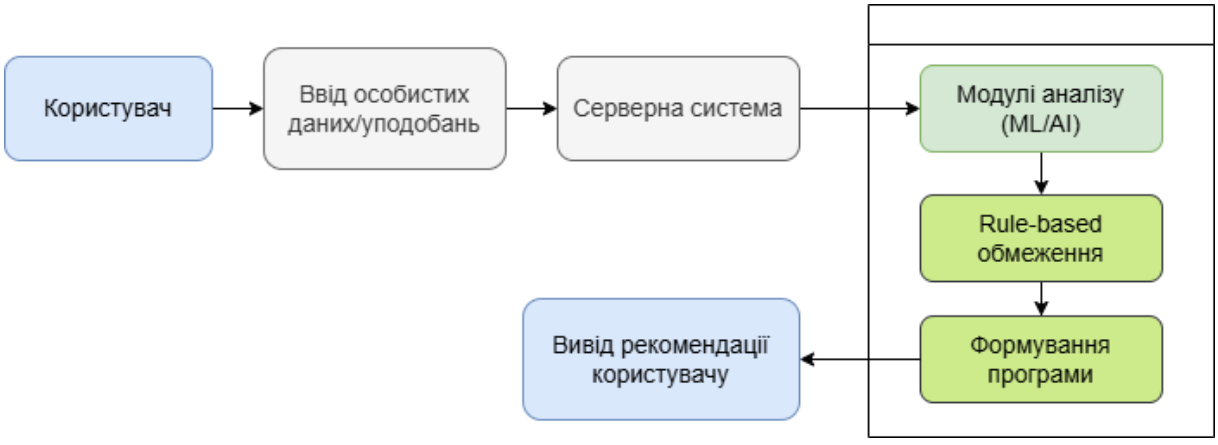


Рисунок 1.1 – Загальний архітектурний потік даних та логіки формування тренувальної програми

Інформація, яку вводить користувач (особисті дані, фізіологічні параметри, уподобання), передається на серверну платформу, де поетапно проходить обробку у модулях AI/ML-аналітики, перевірку згідно експертних rule-based обмежень та формування оптимальної програми тренувань.

Сформовані рекомендації повертаються назад користувачеві у відповідному інтерфейсі. Така побудова забезпечує не лише ефективний вибір програм, а й високий ступінь гнучкості та масштабованості системи.

# 2 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

## 2.1 Формалізація задачі та параметризація користувача

Для забезпечення ефективного математичного моделювання задачі персоналізованого підбору тренувальної програми перш за все необхідною є чітка формалізація параметрів та характеристик користувача. Така формалізація має на меті перетворити набір вихідних даних, що характеризують фізичний стан, параметри тіла та індивідуальні особливості користувача, в обчислювану математичну модель – так званий вектор ознак користувача.

Така векторна модель дає змогу застосовувати різноманітні математичні та алгоритмічні методи для підбору тренувальних програм, використовуючи стандартизовані та формалізовані дані. Для цього запропоновано створити вектор ознак, який буде містити ключові параметри, потрібні для ухвалення рішення щодо вибору програми. Цей вектор ознак може мати такий вигляд:

$$\overrightarrow{x_i} = \begin{bmatrix} \text{вік}_i \\ \text{зріст}_i \\ \text{вага}_i \\ \text{стать}_i \\ \text{рівень}_i \\ \text{ціль}_i \end{bmatrix}, \tag{2.1}$$

де $\text{вік}_i$ – вік користувача, років;

$\text{зріст}_i$ – зріст користувача, см;

$\text{вага}_i$ – маса тіла користувача, кг;

$\text{стать}_i$ – стать користувача;

$\text{рівень}_i$ – рівень фізичної підготовки (початковий, середній, просунутий);

ціль$_i$ – основна мета тренування (оздоровча програма, кардіо, набір маси).

Тренувальні програми математично формалізуються аналогічним вектором параметрів ($\overrightarrow{p_i}$).

## 2.2 Основні алгоритмічні підходи

Для ефективної автоматизації підбору персоналізованих тренувальних програм доцільно застосовувати декілька сучасних алгоритмічних підходів, кожен з яких виконує свою роль у системі рекомендацій.

### 2.2.1 Контент-орієнтована фільтрація

Вибір Контент-орієнтована фільтрація (Content-based filtering) – це класичний алгоритмічний підхід, що базується на оцінці подібності між описом користувача та параметричним описом тренувальних програм на основі їх текстових, числових або категоріальних характеристик. Основною ідеєю цього підходу є виявлення максимальної близькості між параметрами цільового користувача та наявними характеристиками тренувальних програм. Найпоширенішим методом для цього є використання косинусної міри подібності, за допомогою якої можна обчислювати ступінь схожості між числовими векторами характеристик користувачів і програм. Вибір рекомендованої програми виконується за максимальним значенням цієї міри подібності:

$$\cos(\theta) = \frac{\overrightarrow{x_i} \cdot \overrightarrow{y_j}}{|\overrightarrow{x_i}| \cdot |\overrightarrow{y_j}|}, \tag{2.2}$$

де $\overrightarrow{y_j}$ – вектор характеристик програми $p_j$;

$\overrightarrow{x_i} \cdot \overrightarrow{y_j}$ – скалярний добуток векторів;

$|\overrightarrow{x_i}|,\ |\overrightarrow{y_j}|$ – їх норми (евклідова довжина).

Рекомендацією для користувача є та програма, що має найбільшу схожість:

$$p^* = \arg\max_{p_j \in P} \cos(\overrightarrow{x_i}, \overrightarrow{y_j}), \tag{2.3}$$

де $p^*$ – рекомендована програма для користувача;

$P$ – множина всіх тренувальних програм.

### 2.2.2 Колаборативна фільтрація

Колаборативна фільтрація (Collaborative filtering) є алгоритмічним підходом, що використовує історичні дані багатьох користувачів для визначення рекомендаційних переваг окремої особи. Основною ідеєю цього методу є порівняння й аналіз схожих профілів різних користувачів. Якщо користувачі мають схожі характеристики, то ймовірно, що їхні вподобання щодо вибору програм також будуть близькими. Широко застосованою технікою для порівняння користувачів за їх ознаками є кореляція Пірсона, яка дозволяє враховувати й лінійну залежність між параметрами користувачів у рекомендаційних моделях:

$$\mathrm{sim}(u_a, u_b) = \frac{\sum_{l=1}^{k}(x_{a,l} - \bar{x}a)(xb, l - \bar{x}b)}{\sqrt{\sum_{l=1}^{k}(x_{a,l} - \bar{x}a)^2}\sqrt{\sum_{l=1}^{k}(x_{b,l} - \overline{x_b})^2}}, \tag{2.4}$$

де $k$ – кількість ознак;

$x_{a,l},\ x_{b,l}$ – значення l-ї ознаки користувачів $u_a$ та $u_b$;

$\overline{x_a},\ \overline{x_b}$ – середнє значення ознак відповідно.

### 2.2.3 Мультирівнева класифікація

Мультирівнева класифікація є перспективним підходом для вирішення задачі розподілу користувачів на декілька різних класів програм, залежно від їхніх цілей та характеристик (наприклад, силовий тренінг, кардіотренування, оздоровчі програми, програми зі зменшення маси тіла). Цей підхід дозволяє отримувати ймовірності належності конкретного користувача до певної категорії та забезпечує ефективніший підбір програм із заданого класу з відповідним набором характеристик. Ймовірності належності користувача до класу обчислюються за допомогою мультиноміальної логістичної функції, параметри якої визначені під час навчання алгоритму на попередньо зібраних даних:

$$P(C = c\ \vec{x}i) = \frac{e^{\beta c,0+\beta_c^T \vec{x}i}}{\sum c'\ e^{\beta_{c',0}+\beta_{c'}^T \overrightarrow{x_i}}}, \tag{2.5}$$

де $C$ – ймовірнісний клас програм;

$\beta_c$ – параметри для класу $c$, отримані під час навчання;

$c'$ – індекс пробігу по всіх класах.

### 2.2.4 Регресійна модель підбору навантаження

Для точнішого налаштування кількісних параметрів тренувальної програми (наприклад, тривалість вправ, кількість підходів чи повторень, відпочинку, обсягу навантаження) застосовується регресійний підхід. Регресійна модель дозволяє враховувати числові фактори користувачів (вік, вага, рівень підготовленості) й автоматично прогнозувати оптимальні значення таких характеристик. На практиці для цього може використовуватись лінійна регресія, а також складніші регресійні підходи на базі дерев рішень, випадкових лісів, нейронних мереж тощо:

$$y = w_0 + w_1 \cdot \text{вік}_i + w_2 \cdot \text{вага}_i + w_3 \cdot \text{рівень}_i + w_4 \cdot \text{ціль}_i + \epsilon, \quad (2.6)$$

де $y$ – цільовий числовий параметр програми;

$w_0, \dots, w_4$ – вагові коефіцієнти моделі;

$\epsilon$ – випадкова похибка.

2.2.5 Функції втрат для навчання моделей

У процесі навчання класифікаційних і регресійних моделей важливу роль відіграє коректне визначення функцій втрат або помилок. Саме правильно вибрана функція втрат визначає якість роботи моделі та суттєво впливає на її прогностичну точність. Залежно від типу задачі (класифікація чи регресія), використовують різні функції втрат – для задачі класифікації зазвичай застосовується крос-ентропійна функція втрат, тоді як для задач регресії застосовують середньоквадратичну помилку:

$$L_{class} = \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \cdot \log\left(\widehat{y_{i,c}}\right), \quad (2.7)$$

де $N$ – кількість прикладів у вибірці;

$C$ – кількість класів;

$y_{i,c}$ – справжнє значення належності;

$\widehat{y_{i,c}}$ – ймовірність, яку моделює алгоритм.

Для задач регресії – середньоквадратична помилка:

$$L_{reg} = \frac{1}{N} \sum_{i=1}^{N} \left(y_{i,\text{реальне}} - y_{i,\text{прогноз}}\right)^2, \quad (2.8)$$

де $y_{i,\text{реальне}}$ – справжнє значення;

$y_{i,\text{прогноз}}$ – передбачене моделлю.

### 2.2.6 Rule-based та експертні обмеження

Додатково до моделей ML/AI застосовуються експертні правила (наприклад: не призначати силові програми для молодших 15 років, або кардіо для певних медичних обмежень). Дотримання таких правил забезпечує безпеку користувача та підвищує довіру до системи, оскільки дозволяє враховувати індивідуальні обмеження й рекомендації фахівців навіть у складних або нештатних випадках.

## 2.3 Алгоритм комплексної інтеграції

Алгоритм роботи інтелектуальної системи рекомендації (наочно продемонстровано на рисунку 2.1) базується на послідовному застосуванні ряду аналітичних та машинних підходів, що забезпечує послідовну фільтрацію, уточнення та оптимізацію тренувальної програми для кожного користувача. Нижче наведено етапи інтегрованої обробки користувацьких даних і вибору найкращої програми тренування:

– первинний відбір програм за ціллю (ціль$_i$);

– вибір оптимального класу програм з використанням (2.5);

– контентна фільтрація за косинусною схожістю (2.2) і відбір за формулою (2.3);

– додаткове уточнення за колаборативною метрикою (2.4);

– генерація числових параметрів тренування за (2.6);

– перевірка експертних обмежень та коригування програми.

На першому етапі (первинний відбір програм за ціллю користувача) відбувається початкова фільтрація на основі заявлених цілей або побажань користувача (наприклад, схуднення, набір м'язової маси, покращення витривалості). Це дозволяє значно звузити кількість програм, що підлягають

подальшому аналізу, і гарантувати відповідність пропонованих програм основним потребам користувача.

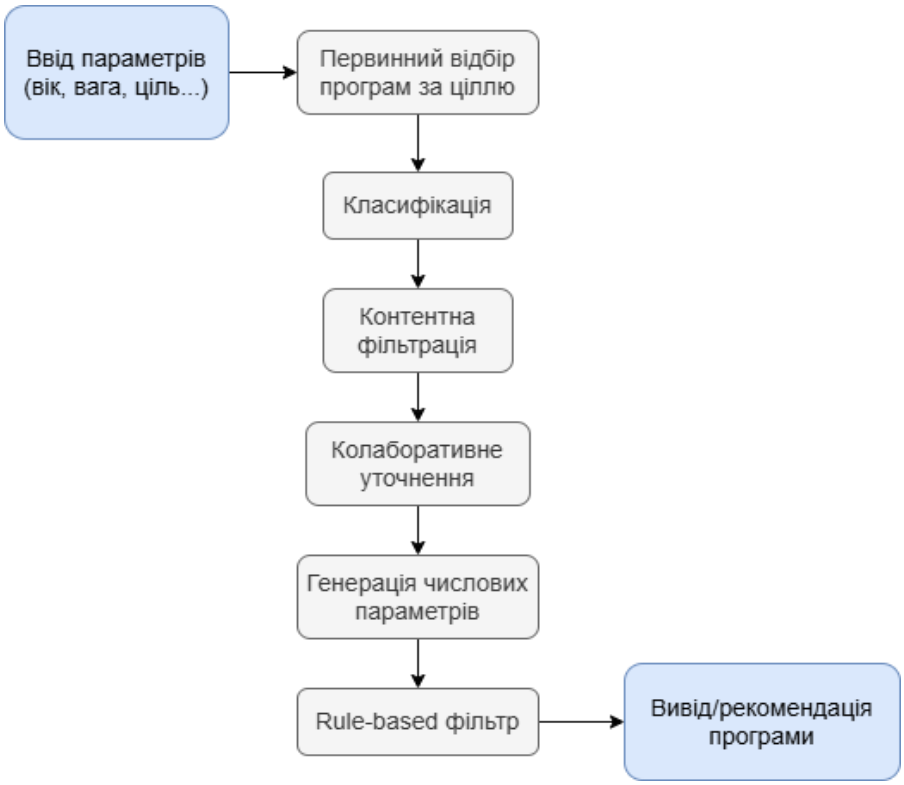


Рисунок 2.1 – Схема етапів підбору тренувальної програми

Наступним є етап вибору оптимального класу програм, де за допомогою класифікаційної моделі (2.5) здійснюється визначення найбільш підходящої категорії або класу програм тренування, що відповідають специфіці профілю, фізичним та антропометричним особливостям користувача. Застосування саме цього підходу дозволяє істотно точніше зорієнтувати рекомендаційну систему у виборі найбільш підходящих тренувальних сценаріїв для подальших етапів уточнення рекомендації.

На третьому етапі застосовується контентна фільтрація програм, які належать до обраного класу, за допомогою розрахунку косинусної схожості (2.2). Даний етап забезпечує кількісний вимір близькості між векторами характеристик користувача та програм тренування. Для забезпечення найкращого результату остаточний відбір програми

проводиться уже за формулою зваженого скору (2.3), яка враховує вагові коефіцієнти основних характеристик і дозволяє отримати найбільш релевантні рекомендації щодо контенту.

Четвертий етап алгоритму – додаткове уточнення з використанням колаборативної метрики (2.4). Цей підхід дозволяє знайти серед інших користувачів найбільш подібний за характеристиками профіль і порівняти його програму тренування з попередньо вибраною програмою поточного користувача. Якщо подібність двох користувачів та програм висока і перевищує певний наперед визначений поріг (як зазначено в алгоритмі, орієнтовне значення тут може бути вище 0,97), то систему рекомендації коригують у бік програми цього «сусіднього» користувача. Таким способом підвищується точність персоналізації і коректність рекомендації.

Наступний, п'ятий етап алгоритму – генерація точних числових характеристик обраної тренувальної програми відповідно до індивідуальних параметрів користувача із застосуванням регресійної моделі (2.6). Ця модель дозволяє спрогнозувати кількісні навантаження, рекомендовану інтенсивність вправ, які оптимально відповідають фізичним можливостям користувача, підвищуючи ефективність і безпечність запропонованої програми.

Заключним етапом є використання правилового модулю (експертна перевірка), завданням якого є оцінка фінальної тренувальної програми щодо можливих протипоказань або інших фізичних та медичних обмежень конкретного користувача (наявні захворювання, індивідуальні особливості). Якщо такі протипоказання виявлені, програма коригується і переобирається у більш безпечному, як правило, оздоровчому напрямі. Це дозволяє мінімізувати ризик отримання травм чи погіршення здоров'я користувача в результаті неправильно підібраної програми тренувань.

# 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

## 3.1 Опис програмного додатку

AI-FIT – це інноваційний веб-застосунок для автоматизованого підбору і генерації персональних тренувальних програм, орієнтованих на максимально гнучкий індивідуальний підхід. Система позиціонується як сучасний помічник для спортсменів, тренерів і всіх, хто відвідує тренажерний зал та прагне підвищити ефективність власних занять. Як зазначають Ricci, Rokach і Shapira, персоналізація є одним з найбільших викликів і водночас найбільших переваг сучасних рекомендаційних систем, забезпечуючи точнішу відповідність потребам користувачів і підвищену мотивацію та ефективність занять [8].

Користувач може розпочати роботу із застосунком одразу після швидкої реєстрації, під час якої задає базові параметри: вік, стать, вагу, ріст, рівень фізичної підготовки та цілі тренувань – наприклад, набір м’язової маси, зниження ваги, робота над витривалістю чи загальне оздоровлення. Ця інформація дозволяє AI-FIT використовувати підходи context-aware recommender systems, враховуючи контекстуальні деталі при генеруванні рекомендацій активностей користувачу, як описано в роботі Adomavicius і Tuzhilin [15]. Всі ці дані зберігаються у профілі й можуть бути скориговані у будь-який момент, якщо змінюються індивідуальні потреби або рівень підготовки користувача.

Після входу до системи користувачі потрапляють до особистого кабінету – структурованої сторінки, де зібрано всю ключову інформацію про персональний профіль. Тут можна переглянути або змінити фізіологічні дані, вибрати або оновити головний напрям фітнес-програми, а також ініціювати генерацію нової тренувальної програми. Весь процес взаємодії максимально інтуїтивний завдяки використанню найкращих практик проектування інтерфейсів enterprise веб-додатків, наведених у посібнику

Мартіна Фаулера Patterns of Enterprise Application Architecture [14], що робить платформу доступною навіть для людей без попереднього досвіду роботи із подібними технологіями.

Однією з основних функцій AI-FIT є автоматична побудова тренувальних комплексів за рахунок сучасних алгоритмів рекомендаційних систем, таких як колаборативна фільтрація (collaborative filtering) [12], що враховують обране спрямування (йога, кросфіт, бодібілдинг, армреслінг, пауерліфтинг, кардіо), стать, рівень підготовки, вагу й ріст користувача. На основі цих алгоритмів користувач миттєво отримує детальний фітнес-план, що містить опис основних днів, конкретні вправи, оптимальні кількості повторень, основні правила виконання й додаткові рекомендації. Завдяки цьому кожен може отримати релевантний перелік фізичних активностей без прямого звернення до фахівців.

Для персоналізації рекомендацій AI-FIT може додатково використовувати платежі гібридних стратегій, яким присвячена окрема увага у статті Бьорка про гібридні рекомендаційні системи (hybrid recommender systems) [10]. Комбінуючи різні техніки та додаючи можливості інтелектуального аналізу, додаток забезпечує високу точність генерованих пропозицій.

Серед ключових можливостей платформи: формування персоналізованого календаря тренувань, де кожен день містить інформацію – назву тренування, його тривалість, тип, а також ім'я тренера, до якого можна звернутися за консультацією. Для самих тренерів система відкриває окремі функції контролю та моніторингу, зокрема, зручний перегляд профілів підопічних, контроль прогресу та експертну підтримку щодо зміни планів тренувань.

AI-FIT реалізований як веб-додаток, побудований за принципами та архітектурними рекомендаціями професійної розробки, описаними в роботі Шарана про Java-проекти та веб-розробку [13]. Використано сучасний Java-фреймворк Spring Boot і надійну реляційну базу даних PostgreSQL.

Інтерфейс побудовано з використанням шаблонного двигуна Thymeleaf та адаптивної верстки. Особлива увага приділена безпеці: персональні дані та процес авторизації реалізовано на основі Spring Security. Архітектура системи модульна, що значно полегшує подальший розвиток та інтеграцію нових компонентів – як розширення списку можливих фітнес-напрямків, так і запровадження нових машинно-навчальних алгоритмів рекомендацій, таких як нейромережеві рішення, запропоновані Covington et al. для побудови глибоких рекомендаційних систем [11].

Отже, унікальність цього застосунку полягає у синергії простоти взаємодії, сучасного дизайну та поглибленої персоналізації завдяки застосуванню сучасних підходів та алгоритмів рекомендаційних систем, що підтверджується успішною практикою розробки аналогічних високотехнологічних рішень [8], [10,] [11], [12], [13], [14], [15]. Всі інтерфейсні рішення лаконічні та інтуїтивні, дозволяючи користувачам максимально швидко орієнтуватися в усіх функціональних частинах системи: головна з інформацією про функціонал, особистий профіль, форма редагування фізичних параметрів, генерація індивідуальних програм тренувань, персональний розклад активностей, сторінки контактів та підтримки.

## 3.2 Архітектура застосунку

AI-FIT реалізований як класичний багаторівневий веб-додаток із чітким розділенням відповідальностей між логічними компонентами. Архітектура побудована за принципами Model-View-Controller (MVC), що дозволяє гнучко розділити рівні представлення даних, бізнес-логіки та взаємодії з користувачем. Використання MVC-підходу повністю відповідає рекомендаціям Мартіна Фаулера, наведеним у Patterns of Enterprise Application Architecture, щодо грамотного поділу відповідальностей між різними рівнями програми [14].

У рамках патерну MVC модель (Model) відповідає за зберігання, передачу та валідацію даних – у нашому застосунку це сутності користувача, тренера, клієнта, тренувального плану, тренування та типу тренування. Модель містить усі ключові поля, які характеризують профіль та фізіологічні параметри користувача, забезпечує цілісність даних у базі та підтримує узгодженість інформації. Такий підхід відповідає перевагам модульного поділу та рекомендаціям щодо організації структур даних для Enterprise-застосунків, запропонованим у роботі Шарана з розвитку веб-додатків на Java [13].

Рівень відображення (View) реалізується через динамічні сторінки на технології Thymeleaf. Кожна сторінка чітко відповідає певному функціоналу: головна сторінка з описом сервісу, форма входу та реєстрації, профіль користувача, генерація програми, календар тренувань, а також контактна інформація. Інтерфейси є інтуїтивними, мають сучасний користувацький дизайн і створені з повною адаптивністю під різні типи пристроїв, що важливо для забезпечення позитивної взаємодії користувачів та відповідає стандартам побудови інтерфейсів сучасних веб-додатків, наведених у літературі [14].

Контролер (Controller) є ключовим елементом зв'язку всієї архітектури: він обробляє запити від користувача, взаємодіє з моделлю та викликає відповідні сервіси. Саме контролер відповідає за логіку аутентифікації, авторизації, перевірки доступу та обробки запитів на генерацію тренувальних програм. Аргументація щодо чіткого розподілу обов'язків та логіки контролерів описана у Patterns of Enterprise Application Architecture Фаулера [14], а також у роботі щодо Java-розробки веб-додатків Шарана [13].

Вся серверна бізнес-логіка інкапсульована в окремому сервісному шарі, який відповідає за конкретні бізнес-завдання: створення плану тренувань, генерування індивідуальної програми, підбір типу тренування згідно з заданими цілями або фізіологічними параметрами. Взаємодія з

постійним сховищем (база даних PostgreSQL) виконана через окремі репозиторії, організовані відповідно до принципів, запропонованих у Patterns of Enterprise Application Architecture, щодо поділу на Data Access Layer та Service Layer [14].

Таким чином, архітектура AI-FIT є модульною та компонентно-орієнтованою: кожен рівень може змінюватися або розширюватися незалежно, що реалізує один із ключових моментів проектування складних інтегрованих інформаційних систем, описаних у згаданих працях [13], [14]. Наприклад, додавання нових видів тренувань або нових рекомендаційних алгоритмів персоналізації не вимагають суттєвого перепроектування всієї системи, що прямо відповідає перевагам сучасних багатошарових Enterprise-застосунків [14], [15]. Єдина логіка аутентифікації та авторизації реалізована із використанням фреймворку Spring Security, що гарантує безпеку інформації та конфіденційність персональних даних, і відповідає сучасним рекомендаціям з безпеки веб-додатків [13].

Ключові переваги обраної архітектури:

– чітке розділення відповідальностей між логічними частинами (MVC);

– можливість масштабування, легке впровадження нових модулів (типів тренувань, ролей, бізнес-логіки);

– простота тестування, підтримки та подальшого розвитку системи;

– в результаті застосунок має сучасну архітектуру, яка одночасно забезпечує надійність, ефективність та гнучкість, що є необхідними умовами для впровадження інноваційних AI-модулів у майбутньому.

## 3.3 Архітектура бази даних

Архітектура бази даних системи AI-FIT розроблена з урахуванням усіх основних сценаріїв роботи застосунку та логіки індивідуального підбору тренувальних програм. Дата-модель включає взаємопов’язані

таблиці для зберігання інформації про користувачів, тренерів, клієнтів (відвідувачів), тренувальні програми, типи занять, а також календар тренувань. Такий підхід відповідає сучасним принципам побудови баз даних для високонавантажених веб-застосунків і рекомендаційних систем [14], [15].

Кожен користувач має унікальний профіль, до якого прив’язуються фізіологічні параметри (зріст, вага, вік, стать), рівень підготовки та ціль тренування. Окрема таблиця відповідає за взаємозв’язок тренера і клієнта, а також дозволяє здійснювати контроль за динамікою персональних планів і відстеженням прогресу. Для кожного тренування зберігається: дата, тип, індивідуальні параметри і відповідальний тренер. Програми тренувань можуть бути унікальними для кожного користувача або сформовані на основі шаблонів із урахуванням поточних параметрів профілю. Таблиця типів тренувань забезпечує можливість гнучко розширювати перелік напрямків (наприклад, додавати нові спортивні дисципліни), що співпадає з підходами розширюваності моделі даних, детально описаними у [13], [14].

Вся структура сховища розроблена для забезпечення масштабованості й швидкого доступу до персональних даних, що особливо важливо при генерації рекомендацій у реальному часі [11]. Детальна логічна схема бази даних представлена на рисунку 3.1.

На цьому рисунку показано основні сутності, їхні атрибути, а також взаємозв’язки між таблицями користувачів, тренерів, клієнтів, програм тренувань, видів занять та розкладу тренувань. Така структура дозволяє легко розширювати функціонал застосунку та підтримувати високі вимоги до персоналізації й безпеки даних, що є ключовими для сучасних фітнес-сервісів [8], [14].

Усі моделі бази даних були переписані у вигляді відповідних Java-класів (ентітей), що відповідають основним сутностям додатку. Це дозволило забезпечити зручну й ефективну роботу з даними на рівні бізнес-

логіки, а також гарантувати коректну взаємодію з реляційною базою даних через ORM (Hibernate/JPA).

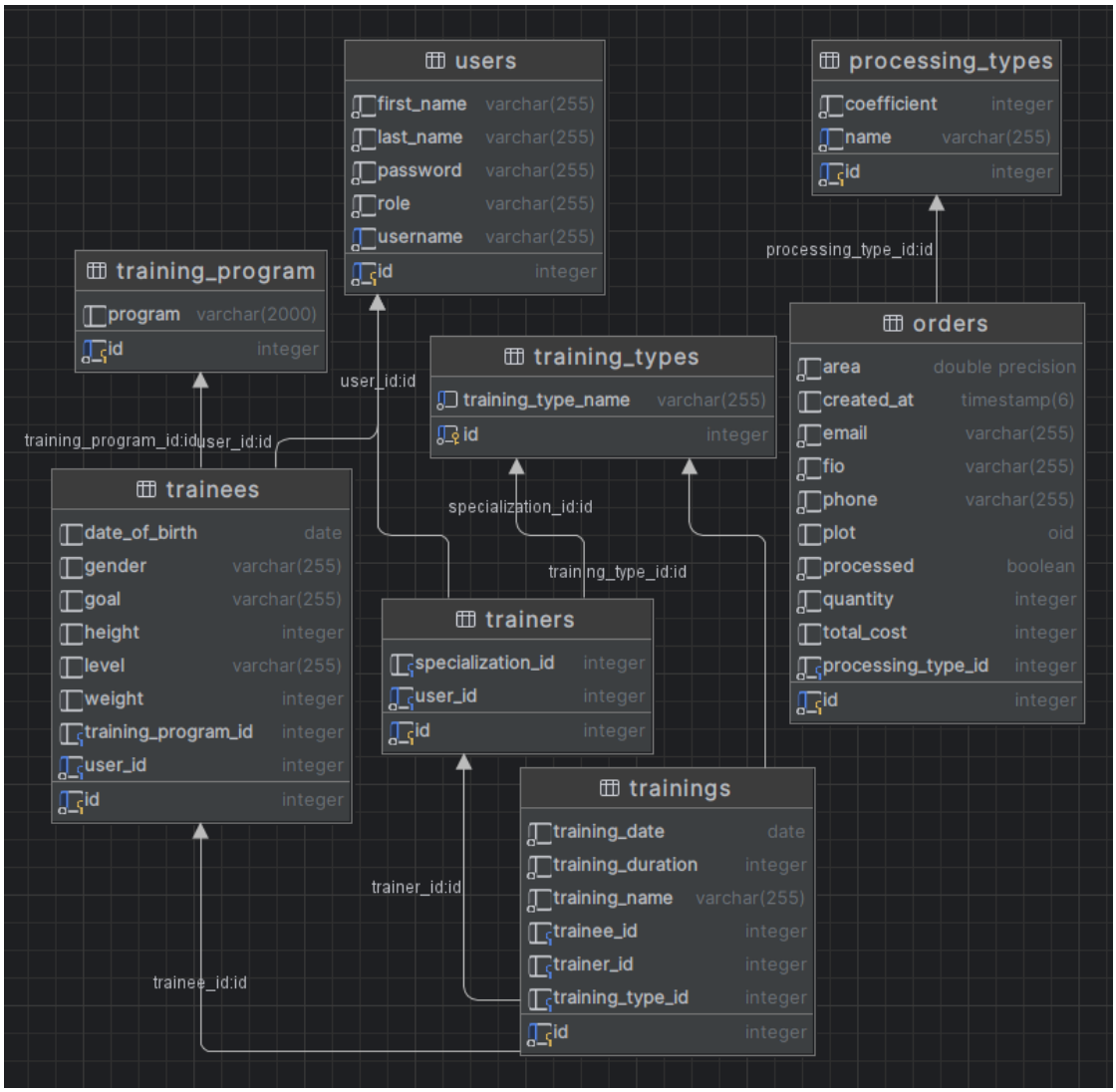


Рисунок 3.1 – Діаграма бази даних системи AI-FIT

Відповідно до рекомендацій офіційної документації Spring Framework та Hibernate [16, 17], всі сутності містять необхідні анотації для зв'язку з таблицями, визначення типів ключів, зв’язків між об'єктами (OneToMany, ManyToOne, тощо) та додаткових атрибутів, що спрощує підтримку і розширення схеми. Повний перелік зв’язків і Java-класів для зберігання користувачів, тренерів, клієнтів, програм, типів тренувань та тренувального розкладу показано на рисунку 3.2.

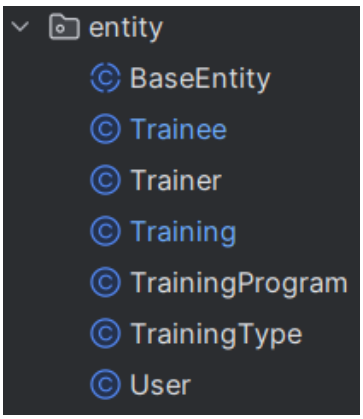


Рисунок 3.2 – Список Java-класів (ентітей) системи AI-FIT

Така модульна модель дає змогу легко масштабувати систему, підвищувати якість персоналізації та інтегрувати новий функціонал без ризику для наявної архітектури.

## 3.4 Реалізація інтелектуальної системи рекомендацій

Для розробки модуля персоналізованих фітнес-рекомендацій у системі AI-FIT було обрано гібридний підхід. Це дозволяє враховувати як індивідуальні параметри користувача (контентна фільтрація), так і досвід схожих клієнтів (колаборативна фільтрація), а також інтегрувати rule-based обмеження для підвищення безпеки. Архітектура основних сервісів для рекомендаційної системи представлена на рисунку 3.3.


Рисунок 3.3 – Структура директорій модулю персоналізованих рекомендацій AI-FIT

У процесі підбору індивідуальної програми дані користувача проходять декілька етапів обробки: спочатку вони перетворюються у числовий вектор, після чого система застосовує послідовно різні алгоритмічні модулі, що дозволяє отримати найбільш релевантну рекомендацію для поточного профілю.

### 3.4.1 Векторизація ознак профілю

Перший етап – приведення усіх описових параметрів користувача до уніфікованої числової форми, що відповідає моделі ознак з теоретичного аналізу (2.1). Це є важливою попередньою процедурою обробки даних (preprocessing), яка безпосередньо впливає на результати майбутнього алгоритмічного аналізу. Саме тому особливу увагу слід приділити вибору максимально точної та ефективної стратегії перетворення вхідних характеристик користувача на числові ознаки.

Існує кілька типів даних, що потребують окремих підходів при кодуванні:

– чисельні (неперервні) параметри – це параметри, що описують користувача за допомогою кількісних характеристик, таких як вік, зріст і вага. Ці дані зазвичай використовуються безпосередньо без додаткових перетворень, хоча іноді доцільно провести нормалізацію або стандартизацію, щоб уникнути дисбалансу під час подібних математичних розрахунків, таких як визначення косинусної подібності;

– категоріальні змінні – це ті, що виражають певну належність користувача до конкретних категорій або класів. Типові категоріальні характеристики у даній задачі – це статева належність (чоловік/жінка), рівень фізичної підготовки (наприклад, початковий, середній або просунутий), а також основна мета тренувань (оздоровчий, кардіо, набір маси тощо). Для цих параметрів заведено використовувати техніку one-hot encoding. Суть цього методу полягає в тому, що з одного категоріального

поля створюється декілька бінарних параметрів. Кожен такий параметр відповідає присутності або відсутності певної категорії у конкретного користувача. Це дозволяє уникнути введення хибних числових зв'язків, які могли б виникнути за неправильного числового кодування категорій через прості числові індекси.

Саме для реалізації зручно структурованої, ефективної та зручної для подальших розрахунків форми запропоновано використовувати спеціально розроблений програмний клас FeatureVector.

Цей клас повністю уніфікує порядок та формат ознак для кожного користувача, дозволяючи потім легко порівнювати різних користувачів або користувачів із програмами (лістинг 3.1).

Лістинг 3.1 – Реалізація класу FeatureVector

```
@Component
public class FeatureVector {
    private final double[] features;
    public FeatureVector(double[] features) {
        this.features = features;    }
    public double[] getFeatures() {
        return features;
    }
    public double cosineSimilarity(FeatureVector other) {
        double dot = 0, normA = 0, normB = 0;
        for (int i = 0; i < features.length; i++) {
            dot += features[i] * other.features[i];
            normA += features[i] * features[i];
            normB       +=       other.features[i]       *
other.features[i];
        }
        if (normA == 0 || normB == 0) return 0;
        return dot / (Math.sqrt(normA) * Math.sqrt(normB));
    }
}
```

Розглянемо детальніше логіку та переваги цього класу:

– масив «features» містить усі числові параметри користувача в уніфікованому форматі. Це можуть бути як вихідні числові показники, так і перетворені за допомогою one-hot encoding категорії. Таким чином, довжина цього вектора буде залежати від кількості початкових ознак та кількості категорій у категоріальних змінних;

– метод getFeatures() дозволяє легко отримати доступ до параметрів і використовувати їх в інших частинах коду або застосовувати інші алгоритмічні підходи без будь-яких ускладнень;

– особливий інтерес представляє метод cosineSimilarity(), який реалізує розрахунок косинусної міри подібності між поточним вектором і будь-яким іншим вектором ознак. Такий підхід дозволяє швидко визначати схожість між користувачами або між користувачем і тренувальною програмою. Завдяки визначенню подібності за косинусною метрикою, ми можемо легко порівнювати дані навіть у випадку, якщо їхні абсолютні значення дуже різняться.

Запропоноване рішення значно спрощує процес майбутньої інтеграції подібних підрахунків та рішень у більш складні алгоритми, забезпечує структурованість коду та полегшує подальше тестування й розширення рішення.

### 3.4.2 Контент-орієнтована фільтрація (Content-based Filtering)

Контент-орієнтований підхід до рекомендацій базується на визначенні подібності між характеристиками користувача і характеристиками рекомендованих об'єктів. У цьому випадку, профіль користувача та кожна існуюча програма представлені у вигляді уніфікованих числових векторів ознак. Основна сутність цього підходу полягає у припущенні, що користувач позитивно сприйме ті програми,

параметри яких максимально відповідають його власним заявленим характеристикам та цілям.

Першим важливим етапом у реалізації контент-орієнтованого підходу є підготовка числового представлення характеристик користувача (вік, вага, зріст, гендер, рівень фізичної підготовки, мета тренувань). Важливо, щоб усі ознаки були представлені в уніфікованому числовому форматі. Як уже було описано раніше, для категоріальних параметрів оптимальним рішенням є використання техніки one-hot encoding, яка гарантує відсутність хибних математичних співвідношень між категоріями. Чисельні параметри (наприклад, вік, вага) можуть бути попередньо нормалізовані чи стандартизовані для уникнення значного дисбалансу у масштабах ознак.

Далі аналогічним чином проводиться побудова векторів ознак і для самих тренувальних програм. Тренувальна програма представлена набором параметрів, таких як «ціль», «тривалість», «складність», «тип тренування». Ці параметри так само обробляються і приводяться у векторну форму в тому ж форматі. Завдяки використанню уніфікованих форм векторів з однаковою кількістю компонентів та категорій, подальше порівняння програм і користувачів стає можливим і зрозумілим з математичної точки зору.

Фінальна стадія цього підходу реалізується шляхом обчислення міри подібності між вектором ознак конкретного користувача та векторами характеристик усіх наявних у базі програм. Одним з найпоширеніших критеріїв для визначення цієї подібності є косинусна схожість. Косинусна міра дозволяє визначити, наскільки близько два вектори розташовані у просторі ознак і порівнювати їх без залежності від абсолютних величин окремих характеристик (лістинг 3.2).

Лістинг 3.2 – Основна логіка content-based рекомендацій

```
@Component
public class ContentBasedRecommender {
    public TrainingProgram recommend(TraineeDto trainee,
List<TrainingProgram> programs) {
```

Продовження лістингу 3.2

```
        FeatureVector userVec =
FeatureVectorUtils.fromTrainee(trainee);
        TrainingProgram best = null;
        double maxSim = -1;
        for (TrainingProgram program : programs) {
            FeatureVector progVec =
FeatureVectorUtils.fromProgram(program);
            double sim =
userVec.cosineSimilarity(progVec);
            if (sim > maxSim) {
                maxSim = sim;
                best = program;
            }
        }
        return best;
    }
}
```

Робота алгоритму полягає у наступному. Спочатку створюється вектор ознак поточного користувача (userVec) на основі його параметрів. Далі в циклі проходять всі тренувальні програми, з кожної програми формується аналогічний вектор ознак (progVec). Після того відбувається обчислення косинусної подібності (метод cosineSimilarity в класі FeatureVector) між цими векторами. У процесі циклу система запам'ятовує ту програму, яка дає максимальне значення подібності із вектором користувача. Отже, програма з максимальною косинусною подібністю вважається оптимальною рекомендацією для конкретного користувача.

Застосування контент-орієнтованого методу має певні переваги. Підхід інтуїтивно зрозумілий та дозволяє гнучко налаштувати систему навіть за обмеженої кількості вихідних даних. Важливою перевагою є також стійкість до «проблеми холодного старту» (cold start problem), коли новому користувачу достатньо заповнити свій профіль, після чого рекомендації уже

стають доступними, тоді як в деяких інших підходах потрібні користувацька історія взаємодій.

Разом з тим, для підвищення ефективності та точності контент-орієнтованої системи можна передбачити додаткові кроки та покращення. Наприклад, окрім прямого використання характеристик програм, можна впроваджувати розширені признакові простори, нові ознаки, комбінації параметрів, вагові коефіцієнти. Також можна додати порогове значення для мінімального рівня схожості, нижче якого рекомендацію система надавати не стане.

Крім того, при розширеній роботі проекту можливо впровадити гібридні підходи, які поєднують переваги як контент-орієнтованих, так і колаборативних методів рекомендацій, що зробить систему гнучкішою і точнішою у підборі оптимальних тренувальних програм.

Запропонована логіка контент-орієнтованої фільтрації, таким чином, дозволяє розраховувати індивідуальні, обґрунтовані та персоналізовані рекомендації для користувачів на основі чітких математично обґрунтованих критеріїв, які зручно реалізувати і розширювати у майбутньому.

### 3.4.3 Колаборативна фільтрація

На відміну від контент-орієнтованого підходу, колаборативна фільтрація не зосереджується виключно на параметрах об'єктів для рекомендацій, натомість враховуючи історію взаємодій і вподобань інших, схожих за характеристиками, користувачів. В основі цього підходу лежить припущення, що користувачі, які були схожими за профілями та вподобаннями у минулому, скоріше за все, матимуть однакові інтереси і у майбутньому.

Колаборативна фільтрація ґрунтується на двох основних етапах. Першим важливим етапом є визначення подібності (або близькості) між профілями різних користувачів. Другий етап – вибір рекомендованого

об’єкта на основі історії вже здійснених виборів найбільш подібних до поточного користувача користувачів.

Запропоноване рішення оцінює близькість користувачів за допомогою адаптованої формули кореляції Пірсона, що дозволяє врахувати усереднені ваги та компенсацію різниці у величинах ознак між користувачами. Кореляція Пірсона є мірою лінійної залежності між двома векторами і приймає значення в діапазоні [-1, 1], де «1» означає ідеально позитивну кореляцію, «-1» – абсолютно негативну, а «0» свідчить про відсутність лінійного взаємозв'язку (лістинг 3.3).

Лістинг 3.3 – Основні методи CollaborativeRecommender

```
@Component
public class CollaborativeRecommender {
    public double pearsonCorrelation(FeatureVector fvA,
FeatureVector fvB) {
        double[] a = fvA.getFeatures();
        double[] b = fvB.getFeatures();
        double meanA =
Arrays.stream(a).average().orElse(0);
        double meanB =
Arrays.stream(b).average().orElse(0);
        double num = 0, denomA = 0, denomB = 0;
        for (int i = 0; i < a.length; i++) {
            num += (a[i] - meanA) * (b[i] - meanB);
            denomA += Math.pow(a[i] - meanA, 2);
            denomB += Math.pow(b[i] - meanB, 2);
        }
        if (denomA == 0 || denomB == 0) return 0;
        return num / Math.sqrt(denomA * denomB);
    }
    public Optional<TraineeDto>
findMostSimilar(TraineeDto target, List<TraineeDto>
others) {
```

Продовження лістингу 3.3

```
        FeatureVector fvTarget =
FeatureVectorUtils.fromTrainee(target);
        double maxSim = -1;
        TraineeDto best = null;
        for (TraineeDto other : others) {
            if(other.getId().equals(target.getId()))
continue;
            double sim = pearsonCorrelation(fvTarget,
FeatureVectorUtils.fromTrainee(other));
            if (sim > maxSim) {
                maxSim = sim;
                best = other;
            }
        }
        return Optional.ofNullable(best);
    }
}
```

Метод pearsonCorrelation приймає два вектори ознак користувачів, після чого обчислює середнє значення ознак для кожного з користувачів окремо. Потім виконується обчислення коваріації між двома векторами характеристик, а також окремих дисперсій для кожного. Отримані значення використовуються для кінцевого розрахунку коефіцієнту кореляції Пірсона. Якщо один із користувачів має нульову дисперсію (тобто усі ознаки мають однакові значення), кореляція визначається як нуль, оскільки неможливо адекватно оцінити лінійну подібність.

Метод findMostSimilar служить для пошуку найбільш схожого користувача до поточного (цільового). Він перебирає усіх інших користувачів та визначає для кожного оцінку схожості шляхом використання методу pearsonCorrelation. В ході перебору зберігається користувач, для якого отримано максимальне значення кореляції, він і є найближчим «сусідом».

Після визначення найближчого сусіда система здійснює рекомендацію програм цього користувача, які мали позитивний досвід та ефективність у минулому. Таким чином, рекомендації формуються на базі фактичного досвіду та реакцій інших користувачів, а не на основі лише описових ознак самої програми.

Для ефективності роботи даного підходу необхідне забезпечення якісного збору і зберігання інформації про історії використання програм користувачами, а саме наявність інформації про вподобання, відвідування тренувальних сесій, рейтинги програм і досягнення поставлених цілей.

### 3.4.4 Прогнозування параметрів

Ключовим завданням ефективної рекомендаційної системи є надання користувачам не лише відповідних типів тренувань, але й максимально точних кількісних параметрів занять (наприклад, кількість підходів, кількість повторень або рекомендоване навантаження). Саме тому додатково вводиться спеціальний модуль прогнозування кількісних параметрів тренувань.

У даній системі запропоновано використовувати класичну модель лінійної регресії для розв'язання задачі прогнозування таких численних параметрів. Лінійна регресія – це статистичний підхід, що дозволяє визначити кількісну залежність між набором вхідних ознак (характеристик користувача) та числовим значенням, яке потрібно прогнозувати. Вона була обрана внаслідок своєї простоти, прозорості обчислень та швидкості роботи, що робить її особливо цінною у контексті мобільних та веб-додатків.

Лістинг 3.4 – Фрагмент коду RegressionModel

```
@Component
public class RegressionModel {
```

Продовження лістингу 3.4

```
    private final double[] w;
    public RegressionModel(double[] w) {
        this.w = w;
    }
    public double predict(FeatureVector fv) {
        double sum = w[0];
        double[] x = fv.getFeatures();
        for (int i = 1; i < w.length; i++) {
            sum += w[i] * x[i-1];
        }
        return sum;
    }
}
```

Метод predict виконує безпосереднє обчислення передбаченого значення на основі переданого у вигляді FeatureVector вектора користувача. Обчислення починається з додавання зміщення (інтерсепта), після чого додаються значення кожної ознаки, зваженої відповідним коефіцієнтом.

Для покращення точності передбачення параметрів у майбутньому можливе застосування кількох технік:

– регуляризація лінійної регресії (наприклад, Ridge або LASSO регресія), що дозволяє боротися з надлишковою складністю моделі та забезпечує її узагальнюваність;

– впровадження поліноміальних характеристик або взаємодій між ознаками для врахування нелінійних зв'язків;

– використання більш складних нелінійних або ансамблевих методів, таких як випадковий ліс (Random Forest), градієнтний бустинг (Gradient Boosting Trees), нейронні мережі, якщо накопичиться достатня кількість навчальних даних.

Окрім цього, варто зазначити про необхідність регулярного перевіряння точності прогнозів та оновлення моделі в міру надходження

нових даних. Підтримка і перепідготовка моделі є важливим процесом забезпечення якості роботи рекомендаційної системи протягом тривалого часу.

Отже, використання модуля прогнозування параметрів тренувань за допомогою лінійної регресії дозволяє створити по-справжньому персоналізовані тренувальні програми, що відповідають можливостям та потребам кожного окремого користувача. Це дозволяє системі не тільки рекомендувати тип тренування, але і пропонувати оптимальні кількісні характеристики вправ, що робить процес тренувань значно ефективнішим і безпечнішим. Крім того, завдяки персоналізованим рекомендаціям користувачі менше ризикують травмуватись або зазнати перенавантаження під час виконання вправ. Водночас це сприяє підвищенню мотивації та покращує загальне сприйняття користувачем рекомендаційної системи.

### 3.4.5 Інтеграція модулів та загальна схема роботи

Після створення та налаштування окремих модулів важливим етапом є їх інтеграція в єдину рекомендаційну систему. Для цієї мети в системі використовується спеціалізований клас-фасад AiFitRecommenderFacade, який відповідає за узгоджену і послідовну роботу всіх розроблених модулів. Він забезпечує єдиний інтерфейс доступу до складних взаємодій, які приховуються від клієнтського коду. Таким чином, кожний модуль працює як окремий компонент, а фасад реалізовує необхідні логічні правила та взаємозв'язок між ними, дотримуючись запропонованої загальної схеми роботи (лістинг 3.6).

Лістинг 3.6 – Загальна інтеграція сервісів

```
@Component
public class AiFitRecommenderFacade {
    private final ProbabilisticClassifier classifier;
private final ContentBasedRecommender contentRecommender;
```

Продовження лістингу 3.6

```
    private final CollaborativeRecommender
collaborativeRecommender;
    private final RegressionModel regression;
    private final RuleBasedExpert expert;
    public AiFitRecommenderFacade(
            ProbabilisticClassifier classifier,
            ContentBasedRecommender contentRecommender,
            CollaborativeRecommender
collaborativeRecommender,
            RegressionModel regression,
            RuleBasedExpert expert
    ) {
        this.classifier = classifier;
        this.contentRecommender = contentRecommender;
        this.collaborativeRecommender =
collaborativeRecommender;
        this.regression = regression;
        this.expert = expert;
    }
    public TrainingProgram getPersonalizedProgram(
            TraineeDto trainee,
            List<TrainingProgram> allPrograms,
            List<TraineeDto> allTrainees
    ) {
        FeatureVector userVector =
FeatureVectorUtils.fromTrainee(trainee);
        double[] probabilities =
classifier.predictProba(userVector);
        int bestClassIdx = maxIdx(probabilities);
        List<TrainingProgram> filteredPrograms =
allPrograms.stream()
                .filter(p -> p.getTypeClassIndex() ==
bestClassIdx)                .toList();
```

Продовження лістингу 3.6

```
        if(filteredPrograms.isEmpty()) {
            filteredPrograms = allPrograms.stream()
                    .filter(p ->
p.getType().equalsIgnoreCase(trainee.getGoal()))
                    .toList();
            if(filteredPrograms.isEmpty()) {
                filteredPrograms = allPrograms;
            }
        }

        TrainingProgram selectedProgram =
contentRecommender.recommend(trainee, filteredPrograms);
        var neighborOpt =
collaborativeRecommender.findMostSimilar(trainee,
allTrainees);
        if (neighborOpt.isPresent()) {
            TraineeDto neighbor = neighborOpt.get();

            TrainingProgram neighborProgram =
neighbor.getTrainingProgram();
            if (neighborProgram != null &&

userVector.cosineSimilarity(FeatureVectorUtils.fromProgra
m(neighborProgram)) > 0.97) {
                selectedProgram = neighborProgram;
            }
        }
        double predictedLoad =
regression.predict(userVector);
selectedProgram.setRecommendedLoad(predictedLoad);
        if (!expert.checkContraindications(trainee,
selectedProgram)) {
            selectedProgram = allPrograms.stream()
                    .filter(p ->
```

Продовження лістингу 3.6

```
p.getType().equalsIgnoreCase("оздоровча"))
                    .findFirst()
                    .orElse(selectedProgram);
        }
       return selectedProgram;
    }
    private int maxIdx(double[] arr) {
        int idx = 0;
        for (int i = 1; i < arr.length; i++) {
            if(arr[i] > arr[idx]) idx = i;
        }
        return idx;
    }
}
```

Загальна логіка роботи фасаду AiFitRecommenderFacade представлена наступними послідовними етапами:

– підготовка даних, інформація у профілі користувача перетворюється у відповідний вектор характеристик (feature vector) для подальших розрахунків моделей;

– класифікація та фільтрація, вектор користувача надходить у ймовірнісний класифікатор (ProbabilisticClassifier), який визначає найбільш підходящу категорію програми. Далі здійснюється початкова фільтрація всіх доступних програм за цією категорією. У разі відсутності таких програм відбувається повторна фільтрація за ключовою ціллю користувача, а за необхідності використовується увесь доступний набір тренувальних програм;

– контент-орієнтований підбір, з урахуванням профілю та характеристик програм відбувається вибір конкретної програми з переліку попередньо відібраних, яка найкраще підходить користувачу;

– колаборативний підхід (персоналізоване уточнення), знаходиться найближчий за характеристиками до нового користувача інший тренувальний профіль (сусід). Якщо програма сусіда вже добре підходить до конкретних характеристик поточного користувача, враховуючи високий коефіцієнт подібності (>0.97), то її встановлюють як цільову програму;

– прогнозування навантаження, за допомогою моделі регресії визначаються точні кількісні параметри (наприклад, навантаження) рекомендованої програми відповідно до характеристик користувача;

– експертна перевіркап, правиловий експерт виконує фінальну перевірку на наявність протипоказань та обмежень у користувача щодо даного типу програми. У разі наявності протипоказань, система автоматично пропонує оздоровчу програму як безпечну альтернативу.

Результат відображення сторінки рекомендованої програми тренувань проілюстроване на рисунку 3.4

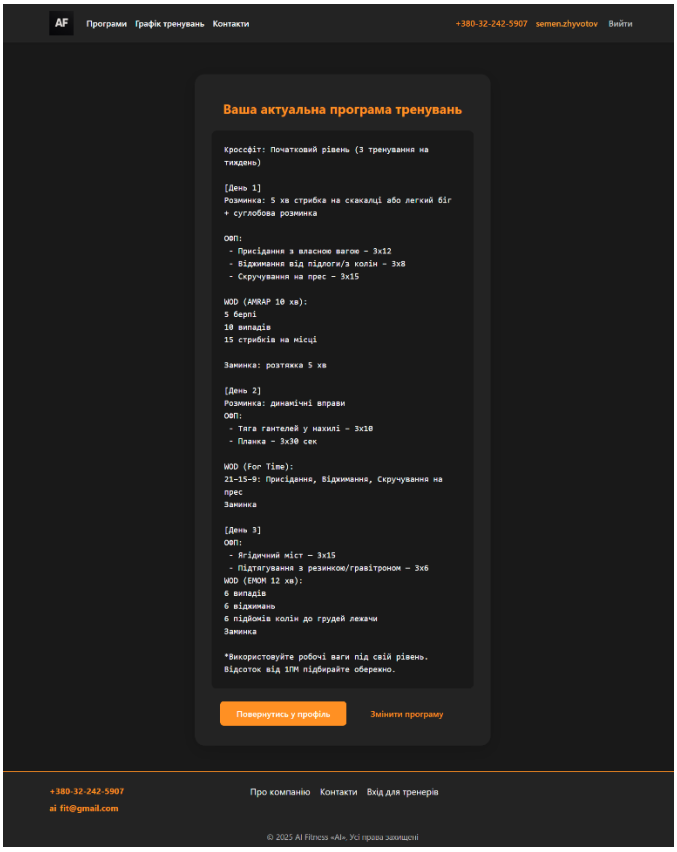


Рисунок 3.4 – Сторінка рекомендованої програми тренувань

AiFitRecommenderFacade надає повноцінний інтегративний інтерфейс рекомендаційної системи, послідовно поєднуючи сильні сторони машинного навчання, експертних оцінок та колаборативної логіки. Це дозволяє покращити якість, точність та персоналізацію генерованих рекомендацій, а також гарантувати високу ефективність та безпечність тренувальних програм, що пропонуються користувачеві за допомогою AiFit.

# ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено та теоретично і практично обґрунтовано концепцію інтелектуальної системи автоматичної генерації тренувальних програм для фітнес-зали на основі сучасних підходів у сфері штучного інтелекту та інформаційних технологій. Проведений аналіз предметної галузі виявив актуальність автоматизації персоналізації фізичних навантажень для різних категорій користувачів, а також основні недоліки існуючих фітнес-програм: шаблонність, відсутність глибокої персоналізації, ігнорування динаміки та індивідуального прогресу.

Було визначено, що оптимальним підходом для вирішення поставленого завдання є гібридна рекомендаційна система, яка поєднує контентно-орієнтовані, колаборативні та rule-based модулі. Теоретично обґрунтовано використання векторної моделі ознак, косинусної метрики для фільтрації, кореляції Пірсона для пошуку «цифрових двійників», а також реалізацію простих моделей класифікації та регресії для визначення величини навантаження та вибору напряму тренування.

У практичній частині роботи впроваджено повноцінний веб-застосунок із сучасним інтерфейсом (адаптивна верстка, логічна навігація), архітектурою на базі Spring MVC та безпечним CRUD-API. Вся серверна логіка розділена за принципом MVC, що дозволяє легко розширювати та модифікувати систему. Для кожного користувача здійснюється зберігання і обробка релевантних фізіологічних та мотиваційних параметрів, формується персональний профіль і автоматично генерується фітнес-програма з урахуванням складності, типу, цілеспрямованості та рекомендацій фахівців.

Описані алгоритми, запропоновані структури класів і приклади коду створюють передумови для впровадження справжніх AI-модулів у майбутньому, з можливістю масштабування, доопрацювання з урахуванням реальних користувацьких даних і нових наукових тенденцій у спортивній

інформатиці. Практична цінність роботи полягає у зручності використання, гнучкості розробленої архітектури, модульності, а також підготовленості до інтеграції новітніх технологій машинного навчання для підвищення ефективності та безпеки фітнес-занять.

Отже, результати дипломної роботи повністю відповідають цілям і завданням дослідження. Запропонована система демонструє наукову й інженерну новизну, вирізняється практичною значущістю та готовністю до впровадження у сучасні цифрові сервіси для спортивних клубів і фітнес-центрів. Реалізована архітектура забезпечує високий рівень персоналізації, гнучкості й масштабованості, що дозволяє ефективно реагувати на потреби користувачів, впроваджувати подальші AI-модулі та сприяє цифровій трансформації фітнес-індустрії.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aggarwal C. C. Recommender Systems: The Textbook. *Springer*, 2016. URL: http://pzs.dstu.dp.ua/DataMining/recom/bibl/1aggarwal_c_c_recommender_systems_the_textbook.pdf. (date of access: 29.04.2025).

2. Building a Personalized Fitness Recommendation Application based on Sequential Information. 2021. URL: https://www.researchgate.net/publication/348910155_Building_a_Personalized_Fitness_Recommendation_Application_based_on_Sequential_Information (date of access: 29.04.2025).

3. Deep Learning for Sensor-based Activity Recognition: A Survey. 2018. URL: https://www.researchgate.net/publication/318392552_Deep_Learning_for_Sensor_based_Activity_Recognition_A_Survey (date of access: 29.04.2025).

4. The Role of Artificial Intelligence in Enhancing Sports Analytics and Training. 2022. URL: https://www.researchgate.net/publication/381774037_The_Role_of_Artificial_Intelligence_in_Enhancing_Sports_Analytics_and_Training. (date of access: 29.04.2025).

5. Title of the Document. Proceedings of the Conference. 2021. URL: https://ieeexplore.ieee.org/document/9802351. (date of access: 29.04.2025).

6. Defynd D. Adidas Using AI: Case Study. URL: https://digitaldefynd.com/IQ/adidas-using-ai-case-study/ (date of access: 29.04.2025).

7. Hive L. H. Fitbod: My AI Physical Trainer. 2021. RL: https://lifehackshive.com/fitbod-my-ai-physical-trainer-61b337cd8fad (date of access: 29.04.2025).

8. Ricci, F., Rokach, L., & Shapira, B. Recommender Systems: Introduction and Challenges. In Recommender Systems Handbook. *Springer*, 2015, pp. 1–34.

9. Koren, Y., Bell, R., & Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer, IEEE,* 2009, pp. 30–37.

10. Burke R. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 2002, pp. 331–370.

11. Covington P., Adams J., Sargin E. Deep Neural Networks for YouTube Recommendations. *Google Research*, 2016. URL: https://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/45530.pdf (date of access: 01.05.2023).

12. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

13. Sharan, S. Mastering Java: An Effective Project-Based Approach including Web Development, Data Structures, GUI Programming, and Object-Oriented Programming (2nd ed.). *Packt Publishing Ltd*, 2020, pp. 50–77.

14. Fowler, M. Patterns of Enterprise Application Architecture. Addison Wesley Professional, 2002, pp. 100–146.

15. Adomavicius, G., & Tuzhilin, A. Context-aware recommender systems. In Recommender Systems Handbook. *Springer*, 2011, pp. 217–253.

16. Spring Framework Reference Documentation, Data Access and Transactions, JPA and Hibernate Support. URL: https://docs.spring.io/spring-framework/reference/data-access.html (date of access: 29.04.2025).

17. Hibernate ORM User Guide. Official documentation. URL: https://docs.jboss.org/hibernate/orm/current/userguide/html_single/Hibernate_User_Guide.html (date of access: 29.04.2025).