МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп'ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ВЕБПЛАТФОРМА З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПЕРСОНАЛІЗАЦІЇ ТРЕНУВАНЬ І ХАРЧУВАННЯ

Виконала: студентка групи 1П-22
Спеціальності
121 Інженерія програмного
забезпечення
Марина ПОНОМАРЕНКО
Керівник:
Валентин ДМИТРЮК

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп'ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувачка циклової комісії КІ та ІТ

_____ Наталя ФАЛЬЧЕНКО

«_____» _____ 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Пономаренко Марині Євгенівні

1. Тема кваліфікаційної роботи Вебплатформа з елементами штучного інтелекту для персоналізації тренувань і харчування

Керівник роботи Дмитрюк Валентин Віталійович, викладач

затверджені наказом закладу фахової передвищої освіти

від «06» жовтня 2025 року № 84/1-у.

2. Строк подання студентом кваліфікаційної роботи 02.06.2026

3. Вихідні дані до кваліфікаційної роботи: мова програмування JavaScript; бібліотека React, інструмент Vite, бібліотеки React Router та Axios для клієнтської частини; середовище Node.js та фреймворк Express для серверної частини; об'єктно-реляційна СКБД PostgreSQL (бібліотека pg); авторизація на основі JWT (jsonwebtoken) та хешування паролів bcrypt; інтеграція з OpenAI API як зовнішнім AI-сервісом; архітектурний стиль REST API; мова опису діаграм UML (діаграми прецедентів, ER-модель, діаграми діяльності).

4. Зміст кваліфікаційної роботи: огляд предметної області (характеристика персоналізованого фітнес-планування та супроводу харчування, аналіз наявних програмних рішень, визначення проблеми й потреб користувачів, постановка завдання); проєктування вебплатформи (функціональні та нефункціональні вимоги, клієнт-серверна архітектура, проєктування бази

даних PostgreSQL та ER-моделі, проєктування REST API, логіка рекомендаційного модуля); реалізація та тестування (серверна частина Node.js/Express, клієнтська частина React, робота з PostgreSQL, генерація тренувального плану та рекомендаційний модуль, тестування основних функцій, демонстрація роботи вебзастосунку).

5. Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	14.10.2025	
2	Розділ 1. Огляд предметної області	09.12.2025	
3	Розділ 2. Проєктування вебплатформи FitAI	10.03.2026	
4	Розділ 3. Реалізація та тестування вебплатформи FitAI	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачці циклової комісії (за 7 днів до захисту)	10.06.2026	

Студент _____ Марина ПОНОМАРЕНКО

Керівник роботи _____ Валентин ДМИТРЮК

АНОТАЦІЯ

Кваліфікаційну роботу присвячено розробленню вебплатформи FitAI з елементами штучного інтелекту для персоналізації тренувань і харчування. У роботі розглянуто предметну область персоналізованого фітнес-планування та ведення харчового журналу, визначено потреби користувачів, сформульовано функціональні й нефункціональні вимоги, спроєктовано клієнт-серверну архітектуру, структуру бази даних PostgreSQL і прикладний програмний інтерфейс REST API.

У межах практичної частини реалізовано вебплатформу, що включає реєстрацію та авторизацію користувача з використанням JWT, роботу з профілем, базу вправ із фільтрацією, генерацію персонального тренувального плану, перегляд плану по днях, позначення тренувань як виконаних, історію тренувань, модуль харчування (добові цілі, журнал, фіксація калорій, білків, жирів, вуглеводів і води, підсумок за день), Dashboard і рекомендаційний модуль.

Клієнтську частину реалізовано засобами React, Vite, React Router та Axios; серверну – на Node.js та Express із базою даних PostgreSQL. Рекомендаційний модуль реалізовано як пояснювану експертну систему на основі правил, що враховує профіль користувача, активний тренувальний план, історію виконання тренувань і харчові показники. Додатково реалізовано інтеграцію з OpenAI API як зовнішнім AI-сервісом для текстового узагальнення рекомендацій. FitAI не є власною нейронною мережею та не виконує машинного навчання на даних користувачів.

Практичне значення роботи полягає у створенні функціональної вебплатформи для персоналізованого планування тренувань, ведення харчового журналу та формування пояснюваних рекомендацій. Рекомендації щодо харчування мають інформаційний характер і не є медичними або дієтологічними призначеннями.

Ключові слова: ВЕБПЛАТФОРМА, FITAI, ПЕРСОНАЛІЗОВАНИЙ ТРЕНУВАЛЬНИЙ ПЛАН, МОДУЛЬ ХАРЧУВАННЯ, РЕКОМЕНДАЦІЙНИЙ МОДУЛЬ, ЕКСПЕРТНА СИСТЕМА НА ОСНОВІ ПРАВИЛ, OPENAI API, REACT, NODE.JS, POSTGRESQL, REST API, JWT.

ANNOTATION

The qualification work is devoted to the development of the FitAI web platform with artificial intelligence elements for the personalization of workouts and nutrition. The work examines the subject area of personalized fitness planning and nutrition logging, defines user needs, functional and non-functional requirements, the client-server architecture, the PostgreSQL database structure, and the REST API design.

The practical part implements a web platform that includes user registration and JWT-based authentication, profile management, an exercise database with filtering, generation of a personalized workout plan, viewing the plan by days, marking workouts as completed, workout history, a nutrition module (daily targets, log, tracking of calories, proteins, fats, carbohydrates and water, daily summary), a Dashboard, and a recommendation module.

The frontend is built with React, Vite, React Router and Axios; the backend is implemented with Node.js and Express using a PostgreSQL database. The recommendation module is implemented as an explainable rule-based expert system that takes into account the user profile, active workout plan, workout history, and nutrition indicators. Additionally, the OpenAI API is integrated as an external AI service for generating textual summaries of recommendations. FitAI is not a proprietary neural network and does not perform self-learning on user data.

The practical value of the work lies in creating a functional web platform for personalized workout planning, nutrition logging, and explainable recommendation support. Nutrition-related recommendations are informational and do not constitute medical or dietary prescriptions.

Keywords: FITAI, WEB PLATFORM, PERSONALIZED WORKOUT PLAN, NUTRITION MODULE, RULE-BASED EXPERT SYSTEM, RECOMMENDATION MODULE, OPENAI API, REACT, NODE.JS, POSTGRESQL, REST API, JWT.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

API – прикладний програмний інтерфейс для взаємодії між клієнтською та серверною частинами.

REST API – архітектурний стиль програмного інтерфейсу, у якому клієнт і сервер обмінюються даними через HTTP-запити до ресурсів.

HTTP / HTTPS – протокол передавання гіпертексту та його захищена версія з шифруванням.

JSON – текстовий формат обміну структурованими даними.

JWT – JSON Web Token – токен для підтвердження автентичності користувача під час доступу до захищених ресурсів.

SQL – мова структурованих запитів до реляційної бази даних.

СУБД – система управління базами даних.

PostgreSQL – реляційна СУБД, використана для зберігання даних вебплатформи FitAI.

CRUD – базові операції над даними: створення, читання, оновлення, видалення.

MVP – мінімально життєздатний продукт – початкова працездатна версія застосунку.

UI / UX – інтерфейс користувача та досвід взаємодії з ним.

CORS – механізм керування доступом до ресурсів між різними джерелами.

Node.js – середовище виконання JavaScript на сервері.

Express – фреймворк Node.js для побудови REST API та маршрутизації.

React – бібліотека JavaScript для побудови клієнтського інтерфейсу.

Vite – інструмент запуску та збирання клієнтської частини.

Axios – бібліотека для виконання HTTP-запитів із клієнта до серверного API.

bcrypt – бібліотека для хешування паролів перед збереженням у базі даних.

pg – бібліотека Node.js для взаємодії з PostgreSQL.

PK / FK – первинний та зовнішній ключі таблиці бази даних.

1:N / M:N – зв’язки «один до багатьох» та «багато до багатьох» між сутностями.

Dashboard – інформаційна панель зі зведеними даними про профіль, план, прогрес, харчування та рекомендації.

Middleware – проміжний обробник HTTP-запиту в серверній частині.

AI – штучний інтелект; у роботі – переважно у складі назви FitAI та для позначення алгоритмічної персоналізації.

FitAI – вебплатформа для персональних тренувальних планів, журналу харчування, Dashboard і пояснюваних рекомендацій.

Рекомендаційний модуль – компонент FitAI, що формує рекомендації на основі правил із профілю, плану, історії та харчових показників.

OpenAI API – зовнішній AI-сервіс, який у FitAI використовується для текстового узагальнення вже сформованих рекомендацій.

ВСТУП

У сучасних умовах цифрові технології активно використовуються для підтримки здорового способу життя, організації фізичної активності, контролю особистого прогресу та ведення базового обліку харчування. Значна кількість користувачів застосовує вебзастосунки та мобільні сервіси для планування тренувань, збереження результатів, перегляду статистики та отримання рекомендацій. Водночас ефективність таких рішень залежить від того, наскільки вони враховують індивідуальні особливості користувача – його рівень підготовки, цілі, доступний режим занять, історію виконання тренувань і фактичні дані щодо харчування.

Актуальність теми зумовлена потребою у зручній вебплатформі, яка не лише дає змогу переглядати окремі вправи чи фіксувати ізольовані показники, а й об’єднує в одному цифровому середовищі тренувальний план, історію виконання, журнал харчування, добові цілі та персоналізовані рекомендації. Користувачі часто мають труднощі із самостійним добором вправ, визначенням послідовності тренувальних днів, підтриманням регулярності занять і контролем базових харчових показників, що створює потребу в єдиному програмному рішенні.

У межах роботи розглядається вебплатформа FitAI, призначена для персоналізації тренувального процесу та інформаційного супроводу харчування. Продукт реалізовано як клієнт-серверну вебсистему: клієнтська частина побудована на React і Vite, серверна – на Node.js та Express, а дані зберігаються у PostgreSQL. Взаємодія між клієнтом і сервером здійснюється через REST API, а доступ до персональних функцій захищено JWT-авторизацією [1; 2; 6; 7; 9; 19].

Назва FitAI є продуктовою. Елементи штучного інтелекту в системі реалізовано комбіновано: через пояснювану експертну систему на основі правил та інтеграцію з OpenAI API як зовнішнім AI-сервісом для генерації текстового узагальнення рекомендацій. Власної нейронної мережі чи самонавчальної моделі машинного навчання на даних користувачів у системі

не реалізовано; OpenAI API не приймає основних рішень, а лише формує зрозумілий текстовий підсумок уже сформованих системою рекомендацій.

Мета роботи – розробити вебплатформу FitAI для персоналізації тренувального процесу, ведення журналу харчування та формування пояснюваних рекомендацій із використанням експертної системи на основі правил і зовнішнього AI-сервісу для текстового узагальнення.

Об'єкт дослідження – процес цифрової персоналізації тренувального процесу та інформаційного супроводу харчування користувача засобами вебплатформи.

Предмет дослідження – методи, моделі даних і програмні засоби розроблення клієнт-серверної вебплатформи для формування персональних тренувальних планів, ведення журналу харчування та надання рекомендацій.

Для досягнення мети необхідно виконати такі завдання:

– проаналізувати предметну область персоналізованого фітнес-планування та ведення харчового журналу;

– розглянути наявні програмні рішення у сфері планування тренувань і обліку харчування;

– сформулювати функціональні та нефункціональні вимоги до вебплатформи FitAI;

– спроєктувати клієнт-серверну архітектуру та структуру бази даних PostgreSQL;

– реалізувати серверну частину (Node.js, Express, PostgreSQL) та клієнтську частину (React, Vite);

– реалізувати реєстрацію, JWT-авторизацію, роботу з профілем, базою вправ і генерацію тренувального плану;

– реалізувати модуль харчування (добові цілі, журнал, підсумок за день) і Dashboard;

– реалізувати рекомендаційний модуль на основі правил та інтеграцію з OpenAI API;

– провести тестування основних функціональних сценаріїв і підготувати демонстрацію роботи.

Окремо слід наголосити на коректному позиціонуванні елементів штучного інтелекту. Назва FitAI є продуктовою, а елементи інтелектуалізації реалізовано комбіновано: через пояснювану експертну систему на основі правил та інтеграцію з OpenAI API як зовнішнім AI-сервісом для текстового узагальнення. Власної нейронної мережі чи навчання на даних користувачів у системі не реалізовано, що дає змогу зберегти контрольованість і пояснюваність рекомендацій та водночас покращити зручність їх сприйняття.

Практичне значення роботи полягає у створенні працездатної вебплатформи, яку можна використовувати як інструмент для організації персонального тренувального процесу, ведення журналу харчування та отримання пояснюваних рекомендацій. Інженерна цінність полягає в реалізації повного циклу розроблення програмного забезпечення – від аналізу предметної області до проєктування, реалізації, інтеграції із зовнішнім AI-сервісом, тестування та демонстрації результату. Рекомендації FitAI мають інформаційний характер і не є медичними або дієтологічними призначеннями.

РОЗДІЛ 1
ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика предметної області

Предметна область персоналізованого фітнес-планування та інформаційного супроводу харчування охоплює процеси формування, виконання й коригування тренувальної активності користувача, а також базове ведення харчових показників. У межах роботи ця область розглядається не як професійна спортивна, медична чи дієтологічна система, а як прикладна сфера створення вебплатформи, що допомагає користувачеві систематизувати тренування, фіксувати основні показники харчування та отримувати зрозумілі інформаційні рекомендації.

Складність для користувача становить не лише саме виконання вправ, а й організація регулярного тренувального процесу та контроль базових харчових показників. Потрібно визначити мету занять, підібрати вправи, розподілити навантаження по днях, фіксувати виконання тренувань, вести історію активності та контролювати добові показники калорій, білків, жирів, вуглеводів і води. За відсутності єдиного цифрового середовища ці дані часто залишаються фрагментованими: тренування плануються окремо, харчування фіксується вручну або не обліковується, а рекомендації не пов’язуються з фактичними діями користувача.

Персоналізоване фітнес-планування відрізняється від універсального набору вправ тим, що план формується з урахуванням параметрів конкретного користувача: рівня підготовки, цілі тренувань, доступного режиму занять, попередньої активності та суб’єктивної оцінки складності. Інформаційний супровід харчування у FitAI полягає у фіксації добових цілей і фактичних показників (калорії, білки, жири, вуглеводи, вода), їх підсумовуванні та порівнянні з цілями. Система не встановлює лікувальних норм і не замінює консультацію спеціаліста.

З погляду інженерії програмного забезпечення розроблення такої системи передбачає підтримку кількох взаємопов’язаних процесів: керування користувачами та захищений доступ до персональних даних; збереження структурованих даних у базі даних; серверну логіку обробки запитів клієнта та взаємодії з базою; клієнтський інтерфейс для виконання основних сценаріїв без зайвої складності; рекомендаційний механізм, що пов’язує накопичені дані з практичними підказками. Саме поєднання цих процесів у межах єдиної платформи й визначає прикладну цінність FitAI.

Основними суб’єктами предметної області є користувач та програмна система. Об’єктами взаємодії виступають профіль користувача, вправа, тренувальний план, день плану, факт виконання тренування, історія тренувань, добові цілі харчування, запис журналу харчування, підсумок за день і рекомендація. Типовий сценарій починається з реєстрації та авторизації, після чого користувач заповнює профіль, переглядає вправи, формує план, виконує тренування, веде журнал харчування та аналізує Dashboard. Важливою вимогою є поєднання простоти використання з достатнім рівнем персоналізації, а також пояснюваність рекомендацій, щоб користувач розумів, які саме дані вплинули на конкретну пораду.

Окремою проблемою предметної області є фрагментованість даних. У повсякденному режимі інформація про цілі, вправи, тренувальні плани, виконані заняття та харчові показники часто зберігається окремо або взагалі не фіксується, що ускладнює аналіз прогресу та прийняття подальших рішень. Вебплатформа для персоналізації тренувань і харчування має об’єднувати ці дані в межах єдиної інформаційної системи, де кожен елемент пов’язаний із профілем конкретного користувача.

Характерною є й потреба у пояснюваності рекомендацій: користувач має розуміти, чому система пропонує певні дії. У межах FitAI рекомендаційний модуль реалізовано як пояснювану експертну систему на основі правил – базові рекомендації формуються через алгоритмічну обробку визначених параметрів (профіль, активний план, історія виконання, добові цілі та записи

журналу харчування), а не за допомогою нейронної мережі чи моделі машинного навчання. З погляду інженерії програмного забезпечення розроблення такої системи потребує послідовного проходження етапів життєвого циклу ПЗ: аналізу предметної області, визначення вимог, проєктування, реалізації, тестування та оцінювання результатів [29; 30].

1.2 Аналіз наявних програмних рішень

Для визначення функціональних орієнтирів майбутньої вебплатформи проведено порівняльний аналіз наявних рішень у сфері фітнес-трекінгу, персоналізації тренувань і обліку харчування. Метою є виявлення типових підходів до організації функціональності, персоналізації, збереження історії активності, ведення харчового журналу та формування рекомендацій. До аналізу включено кілька груп рішень: застосунки для обліку харчування, екосистеми фітнес-пристроїв, сервіси GPS-трекінгу, застосунки з персоналізованими планами та бібліотеки тренувань.

Сервіси на зразок MyFitnessPal орієнтовані переважно на облік харчування, калорій і макронутрієнтів, проте не є основним інструментом для формування тренувального плану. Екосистеми фітнес-пристроїв (Google Fit, Fitbit, Apple Fitness) забезпечують збір даних про активність, але залежать від наявності пристроїв. Сервіси спортивного трекінгу (Strava) орієнтовані на фіксацію активностей на відкритому повітрі та соціальну взаємодію. Застосунки з персоналізованими планами (Freeletics, Fitbod) мають вищу міру персоналізації, проте є комерційними із закритою логікою. Бібліотеки тренувань (Nike Training Club) надають готовий контент, але персоналізація в них зводиться до вибору програми, а не побудови плану на основі структурованого профілю. Порівняння рішень наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння наявних програмних рішень

Рішення	**Основне призначення**	**Тренування**	**Харчування**	**Персоналізація**
MyFitnessPal	Облік харчування й калорій	Допоміжний	Розвинений журнал	Через харчові цілі

Google Fit / Fitbit	Моніторинг активності	Через пристрої	Залежить від інтеграцій	Залежить від пристроїв
Strava	GPS-трекінг активностей	Сильний трекінг	Не є фокусом	Активність і соціальні функції
Freeletics / Fitbod	Персональні тренування	Розвинені плани	Допоміжне	Висока, але закрита
Nike Training Club	Бібліотека тренувань	Готові програми	Не є фокусом	Через вибір програм
FitAI	Персоналізація тренувань і харчування	Генерація плану, історія	Журнал, цілі, підсумок	Правила + AI-узагальнення

Розглянемо виокремлені групи детальніше. Першу групу становлять сервіси обліку харчування (наприклад, MyFitnessPal). Їхньою сильною стороною є структуроване ведення харчового щоденника, контроль добових показників і накопичення історії. Водночас вони не є основним інструментом саме для формування тренувального плану, оскільки їхній центральний сценарій пов’язаний передусім із харчуванням та енергетичним балансом.

Другу групу утворюють екосистеми фітнес-пристроїв і пов’язаних застосунків (Google Fit, Fitbit, Apple Fitness). Вони забезпечують збір і відображення даних про активність, сон, пульс та інші показники за наявності відповідного пристрою. Перевагою є зручне накопичення історії та підказки на основі реальних даних, а обмеженням – залежність від екосистеми пристроїв і менша зручність для користувача, якому потрібен простий вебінструмент без додаткового обладнання.

Третю групу становлять сервіси спортивного трекінгу (Strava), орієнтовані на фіксацію активностей на відкритому повітрі та соціальну взаємодію. Вони мають сильні механізми ведення історії та аналізу прогресу, проте менш придатні як універсальний інструмент для початкового формування плану та ведення журналу харчування в одному інтерфейсі. Четверту групу формують застосунки з персоналізованими планами (Freeletics, Fitbod), що адаптують тренування до цілей і рівня підготовки; їхнім обмеженням є закрита комерційна логіка, переважно мобільний формат та обмежена прозорість алгоритмів. Окрему групу становлять бібліотеки тренувань (Nike Training Club) із готовим контентом, де персоналізація

зводиться до вибору програми, а не побудови плану на основі структурованого профілю та історії виконання.

Аналіз показує, що наявні рішення мають розвинені окремі напрями, проте часто спеціалізуються на одному сценарії – обліку харчування, зборі даних із пристроїв, GPS-трекінгу, готових програмах або персоналізації силового тренування. Для користувача-початківця це створює фрагментованість даних. FitAI доцільно позиціонувати як навчальну клієнт-серверну вебплатформу, у якій в межах одного середовища поєднано базові, але практично важливі сценарії: формування персонального плану, роботу з вправами, фіксацію виконання, ведення журналу харчування, Dashboard і пояснюваний рекомендаційний модуль.

1.3 Визначення проблеми та потреб користувачів

На основі аналізу предметної області та наявних рішень визначено основну прикладну проблему: відсутність простого єдиного інструменту, який поєднує персоналізоване тренувальне планування, фіксацію виконаних тренувань, базовий облік харчування та пояснювані рекомендації в межах одного вебсередовища. Користувачі стикаються з труднощами добору вправ, побудови структури плану по днях, контролю регулярності, фрагментованістю харчових даних і непрозорістю рекомендацій у наявних сервісах.

Користувачі, які прагнуть організувати власний тренувальний процес, стикаються з кількома типовими труднощами. По-перше, складно самостійно підібрати вправи відповідно до рівня підготовки та мети. По-друге, навіть за наявності окремих вправ бракує зрозумілої структури плану по днях. По-третє, без фіксації виконаних тренувань важко оцінити регулярність і прогрес. По-четверте, дані щодо харчування часто ведуться окремо або не фіксуються взагалі. По-п’яте, рекомендації у фітнес-сервісах не завжди прозорі, оскільки користувач не бачить, які саме дані вплинули на результат. Саме на усунення цих труднощів спрямовано розроблення FitAI.

FitAI спрямований на вирішення цих проблем через створення клієнт-серверної платформи, у якій профіль користувача, тренувальний план, історія

виконання, журнал харчування та рекомендації пов’язані між собою. Основою персоналізації є профіль користувача; на його основі система генерує план, відображає його по днях, дозволяє позначати тренування виконаними та накопичує історію. Модуль харчування дає змогу працювати з добовими цілями, додавати записи, фіксувати калорії, білки, жири, вуглеводи та воду й переглядати підсумок за день. Рекомендації формуються пояснюваною експертною системою на основі правил, а OpenAI API використовується лише для текстового узагальнення.

Узагальнено потреби користувачів FitAI можна згрупувати за кількома напрямами:

– створення облікового запису та безпечний доступ до персональних даних;

– заповнення профілю користувача для персоналізації;

– перегляд бази вправ і фільтрація вправ за основними параметрами;

– генерація персонального тренувального плану та перегляд його по днях;

– позначення тренування як виконаного та перегляд історії виконаних тренувань;

– створення або редагування добових цілей харчування;

– ведення журналу харчування та фіксація калорій, білків, жирів, вуглеводів і води;

– перегляд підсумку харчування за день і зведеної інформації на Dashboard;

– отримання пояснюваних рекомендацій та їх текстового AI-узагальнення.

Окремою потребою є ведення базового журналу харчування, що реалізується через модуль харчування: робота з добовими цілями, додавання записів, фіксація калорій і макронутрієнтів та перегляд підсумку за день. Цей модуль не має призначення медичного або дієтологічного сервісу – його функція полягає в інформаційній підтримці користувача та зручному

структуруванні введених даних. Не менш важливою є потреба в отриманні зрозумілих рекомендацій. Обрано саме rule-based підхід, оскільки він контрольований, зрозумілий і придатний для тестування в межах навчального продукту, а OpenAI API залучається лише для формування текстового узагальнення вже сформованих рекомендацій.

1.4 Постановка завдання до кваліфікаційної роботи

На основі проведеного аналізу основне завдання роботи сформульовано як розроблення вебплатформи FitAI з елементами штучного інтелекту для персоналізації тренувань і харчування. Продукт має забезпечити користувачеві можливість створити обліковий запис, пройти авторизацію, заповнити профіль, отримати персоналізований план, переглядати його по днях, фіксувати виконання, вести журнал харчування, працювати з добовими цілями, переглядати підсумок за день та отримувати пояснювані рекомендації. Вебплатформа реалізується як клієнт-серверна система з React frontend, Node.js / Express backend і PostgreSQL.

Узагальнену взаємодію користувача із системою відображено на діаграмі прецедентів (рис. 1.1), яка показує основні сценарії використання вебплатформи.

Рисунок 1.1 – Діаграма прецедентів вебплатформи FitAI

Для досягнення мети необхідно реалізувати такі основні функціональні можливості: реєстрацію та JWT-авторизацію із захистом приватних сторінок; створення й редагування профілю; зберігання та фільтрацію бази вправ; генерацію персонального тренувального плану та перегляд його по днях; позначення тренування виконаним і збереження історії; створення й редагування добових цілей харчування; ведення журналу харчування з фіксацією калорій, білків, жирів, вуглеводів і води; формування підсумку за день; відображення Dashboard; формування рекомендацій на основі правил та генерацію текстового AI-узагальнення через OpenAI API.

Рекомендаційний модуль описується як пояснювана експертна система на основі правил, а не як система машинного навчання. Інтеграція з OpenAI API має допоміжний характер – використовується для текстового узагальнення вже сформованих рекомендацій. До нефункціональних вимог належать зручність використання, надійність, безпека, підтримуваність і розширюваність. Безпека забезпечується хешуванням паролів, JWT-авторизацією та розмежуванням доступу; секретні ключі, зокрема ключ OpenAI API, зберігаються лише на серверній стороні у змінних середовища. FitAI є навчальним інженерним продуктом і не описується як медична, дієтологічна або професійна спортивна система.

Цільовою аудиторією вебплатформи є користувачі, які прагнуть систематизувати тренувальний процес і базовий облік харчування без необхідності самостійно складати повну програму занять або вести показники в різних сервісах. Для початківців важливо отримати зрозумілу структуру тренувального процесу, а для користувачів із певним досвідом – можливість відстежувати виконання плану, бачити історію активності та працювати з журналом харчування. Тому FitAI має бути не довідником вправ, а прикладною вебплатформою, що поєднує зберігання користувацьких даних, роботу з базою вправ, генерацію плану, ведення харчового журналу та пояснюваний рекомендаційний механізм.

Очікуваним результатом виконання роботи є працездатна вебплатформа FitAI, яка демонструє повний цикл створення програмного продукту: аналіз предметної області, формулювання вимог, проєктування архітектури та бази даних, реалізацію клієнтської й серверної частин, створення REST API, реалізацію тренувального планування, модуля харчування, Dashboard, рекомендаційного модуля, інтеграції з OpenAI API та перевірку основних сценаріїв роботи. При цьому система свідомо не позиціонується як медичний, дієтологічний чи професійний спортивний інструмент, а елементи штучного інтелекту описуються відповідно до фактичної реалізації – як поєднання rule-based логіки та зовнішнього сервісу текстового узагальнення.

Висновки до розділу 1

У першому розділі було розглянуто предметну область вебзастосунків для підтримки тренувального процесу, контролю фізичної активності та ведення харчових показників. Визначено, що цифрові рішення у сфері фітнесу дають змогу користувачам зручніше планувати тренування, контролювати регулярність занять, фіксувати прогрес і працювати з даними щодо харчування.

Проведений аналіз наявних програмних рішень показав, що більшість фітнес-застосунків і сервісів орієнтовані або на тренування, або на харчування, або на окремі елементи самоконтролю. Через це користувачеві часто доводиться працювати з кількома різними інструментами. Така фрагментація ускладнює повсякденне використання цифрових засобів підтримки здорового способу життя.

У межах аналізу було визначено основні потреби цільових користувачів: можливість створення персонального профілю, отримання тренувального плану, перегляд вправ, фіксація виконаних тренувань, ведення історії, облік харчових показників і отримання рекомендацій. Також встановлено, що система повинна бути зрозумілою для користувача, не вимагати складної технічної підготовки та забезпечувати доступ до основних функцій через вебінтерфейс.

За результатами огляду предметної області сформульовано завдання кваліфікаційної роботи – розробити вебплатформу FitAI, яка поєднує функції роботи з профілем користувача, базою вправ, персональним тренувальним планом, історією тренувань, модулем харчування, Dashboard, рекомендаційним модулем і текстовим AI-узагальненням. При цьому рекомендації мають формуватися на основі зрозумілих правил, а зовнішній AI-сервіс використовується лише для текстового узагальнення вже підготовлених рекомендацій.

Таким чином, результати першого розділу підтверджують актуальність створення вебплатформи FitAI та визначають функціональні напрями, які мають бути реалізовані під час проєктування й розроблення системи.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ FITAI

2.1 Функціональні та нефункціональні вимоги до системи

Формування вимог є одним із ключових етапів проєктування програмного забезпечення, оскільки саме вимоги визначають очікувану поведінку системи, її функції, обмеження та якісні характеристики [29; 30]. FitAI розглядається як клієнт-серверна вебплатформа для персоналізації тренувань і харчування. Основною роллю є зареєстрований користувач, який створює профіль, працює з тренувальним планом, фіксує виконані тренування, веде журнал харчування та отримує рекомендації. Клієнтська частина реалізується засобами React, серверна – на Node.js та Express, дані зберігаються у PostgreSQL.

Функціональні вимоги визначають дії, які має виконувати система, та можливості, які вона надає користувачеві. Повний перелік функціональних вимог до FitAI наведено в таблиці 2.1.

Таблиця 2.1 – Функціональні вимоги до вебплатформи FitAI

Код	Назва вимоги	Важливість
FR-01	Реєстрація нового користувача	Критична
FR-02	Авторизація користувача з використанням JWT [14; 16; 23]	Критична
FR-03	Обробка помилок під час реєстрації та авторизації	Висока
FR-04	Захист доступу до приватних сторінок користувача	Критична
FR-05	Створення та перегляд профілю користувача	Критична
FR-06	Редагування профілю користувача	Висока
FR-07	Перегляд бази вправ	Критична
FR-08	Фільтрація вправ за параметрами	Висока
FR-09	Перегляд інформації про окрему вправу	Середня
FR-10	Генерація персонального тренувального плану	Критична
FR-11	Збереження згенерованого тренувального плану	Критична
FR-12	Перегляд тренувального плану по днях	Критична
FR-13	Перегляд вправ у межах конкретного дня плану	Висока
FR-14	Позначення тренування як виконаного	Критична

Код	Назва вимоги	Важливість
FR-15	Збереження історії виконаних тренувань	Критична
FR-16	Перегляд історії виконаних тренувань	Висока
FR-17	Створення або первинне формування добових цілей харчування	Критична
FR-18	Редагування добових цілей харчування	Висока
FR-19	Додавання запису до журналу харчування	Критична
FR-20	Перегляд журналу харчування за день	Висока
FR-21	Формування підсумку харчування за день	Критична
FR-22	Відображення сторінки NutritionPage	Критична
FR-23	Відображення Dashboard з основною інформацією	Критична
FR-24	Відображення поточного тренувального прогресу	Висока
FR-25	Відображення короткого підсумку харчування на Dashboard	Висока
FR-26	Формування рекомендацій на основі правил	Критична
FR-27	Урахування історії тренувань у рекомендаціях	Висока
FR-28	Урахування харчових показників у рекомендаціях	Висока
FR-29	Генерація текстового AI-узагальнення рекомендацій	Критична
FR-30	Коректна робота системи без AI-узагальнення	Висока
FR-31	Обмін даними між клієнтською та серверною частинами через REST API	Критична
FR-32	Збереження даних у PostgreSQL	Критична
FR-33	Валідація основних даних, що вводяться користувачем	Висока

Нефункціональні вимоги визначають якісні характеристики системи. Для FitAI важливими є продуктивність, зручність використання, безпека, конфіденційність, надійність, підтримуваність, тестованість і коректна робота із зовнішнім AI-сервісом. Основні нефункціональні вимоги наведено в таблиці 2.2.

Таблиця 2.2 – Нефункціональні вимоги до вебплатформи FitAI

Код	Атрибут якості	Нефункціональна вимога
NFR-01	Зручність використання	Інтерфейс вебплатформи має бути зрозумілим для користувача та забезпечувати доступ до основних функцій без спеціальної технічної підготовки.

Код	Атрибут якості	Нефункціональна вимога
NFR-02	Логічність навігації	Основні сторінки системи – Dashboard, профіль, вправи, тренувальний план, історія, NutritionPage і рекомендації – мають бути логічно пов’язані між собою.
NFR-03	Продуктивність клієнтської частини	Основні сторінки інтерфейсу мають відкриватися без критичних затримок у локальному або навчальному демонстраційному середовищі.
NFR-04	Продуктивність API	Серверна частина має обробляти основні запити користувача без критичних затримок у звичайному сценарії використання MVP.
NFR-05	Безпека авторизації	Доступ до приватних сторінок і персональних даних користувача має здійснюватися лише після авторизації з використанням JWT.
NFR-06	Захист облікових даних	Паролі користувачів не повинні зберігатися у відкритому вигляді; для їх обробки має використовуватися хешування.
NFR-07	Конфіденційність даних	Користувач повинен мати доступ лише до власного профілю, тренувального плану, історії тренувань, харчових записів, добових цілей і рекомендацій.
NFR-08	Цілісність даних	Дані в PostgreSQL мають зберігатися узгоджено, із дотриманням зв’язків між користувачами, профілями, вправами, планами, історією, харчуванням і рекомендаціями.
NFR-09	Безпека конфігурації	Секретні значення, зокрема параметри підключення до бази даних, JWT secret і OpenAI API key, мають зберігатися на серверній стороні у змінних середовища.
NFR-10	Обробка помилок	Система має коректно обробляти типові помилки: некоректні облікові дані, відсутність профілю, відсутність активного плану, помилки API або недоступність зовнішнього AI-сервісу.
NFR-11	Відмовостійкість щодо OpenAI API	Недоступність OpenAI API не повинна блокувати основні функції FitAI; базові рекомендації на основі правил мають залишатися доступними.
NFR-12	Пояснюваність рекомендацій	Рекомендації мають формуватися на основі зрозумілих правил, пов’язаних із профілем, активним планом, історією тренувань і харчовими показниками.
NFR-13	Коректність AI-позиціонування	FitAI не повинен описуватися як власна нейронна мережа або самонавчальна ML-модель; OpenAI API використовується як зовнішній сервіс для текстового узагальнення.
NFR-14	Інформаційний характер харчового модуля	Дані й рекомендації щодо харчування мають подаватися як інформаційні підказки, а не як медичні або дієтологічні призначення.
NFR-15	Супроводжуваність	Код клієнтської та серверної частин має бути структурований за відповідальністю: сторінки й компоненти React, маршрути Express, контролери, сервіси та модулі роботи з базою даних.

Код	Атрибут якості	Нефункціональна вимога
NFR-16	Розширюваність	Архітектура системи має дозволяти подальше додавання нових функцій без повної переробки основної структури вебплатформи.
NFR-17	Тестованість	Основні сценарії роботи системи мають бути придатні до ручної функціональної перевірки через інтерфейс користувача та, за потреби, через API-запити.

Нефункціональні вимоги до FitAI сформульовано з урахуванням того, що система є навчальним інженерним MVP, а не промисловою високонавантаженою платформою. Тому вимоги до продуктивності, надійності та сумісності оцінюються в межах локального або демонстраційного середовища, необхідного для перевірки основних сценаріїв роботи.

Особливу увагу приділено безпеці авторизації та захисту персональних даних користувача. Доступ до приватних функцій системи має здійснюватися лише після авторизації, паролі не повинні зберігатися у відкритому вигляді, а секретні параметри backend, зокрема ключ OpenAI API, мають зберігатися у змінних середовища.

Окремо визначено вимоги до коректного позиціонування рекомендаційного модуля та AI-узагальнення. FitAI не є власною нейронною мережею і не виконує машинного навчання на даних користувачів. Основна логіка рекомендацій реалізується як пояснювана система на основі правил, а OpenAI API використовується як зовнішній сервіс для текстового узагальнення вже сформованих рекомендацій.

Модуль харчування має інформаційний характер. Він призначений для фіксації введених користувачем показників, формування добового підсумку та порівняння фактичних значень із добовими цілями. Система не є медичною або дієтологічною платформою, а її рекомендації не повинні подаватися як призначення спеціаліста.

2.2 Загальна архітектура вебплатформи FitAI

Архітектуру FitAI побудовано за клієнт-серверним принципом із розподілом на чотири логічні рівні. Перший – presentation layer, клієнтський інтерфейс на React, що відповідає за відображення сторінок, форм, списків, тренувальних планів, історії, харчових даних, Dashboard і рекомендацій. Другий – application layer, серверна частина на Node.js / Express, що реалізує маршрути API, авторизацію, генерацію планів, роботу модуля харчування, формування rule-based рекомендацій і підготовку даних для AI-узагальнення. Третій – data layer, база даних PostgreSQL для довготривалого зберігання. Четвертий – external AI service layer, інтеграція з OpenAI API для текстового узагальнення рекомендацій [24; 25].

Клієнтську частину реалізовано з використанням React і Vite [1; 2; 3; 4; 5]. React забезпечує компонентний підхід, за якого сторінки та елементи поділяються на незалежні компоненти, що спрощує супровід і дає змогу динамічно оновлювати інтерфейс без повного перезавантаження. Для переходів між сторінками використовується React Router, а для обміну даними із сервером – Axios, який надсилає HTTP-запити до REST API. На рівні frontend реалізовано сторінки реєстрації та авторизації, Dashboard, профілю, бази вправ, тренувального плану, історії тренувань, NutritionPage та блоки рекомендацій.

Основними модулями вебплатформи є модуль автентифікації (реєстрація, вхід, перевірка JWT-токена), модуль профілю (персональні параметри для персоналізації), модуль вправ (база та фільтрація), модуль тренувальних планів (генерація та перегляд по днях), модуль історії тренувань (фіксація виконання й прогрес), модуль харчування (добові цілі, журнал, підсумок за день), модуль Dashboard (зведена інформація), рекомендаційний модуль (правила) і модуль AI-узагальнення (звернення до OpenAI API). Такий поділ відповідальності робить структуру системи зрозумілою та придатною до розширення.

Серверну частину побудовано на Node.js та Express [6; 7; 8]. Вона приймає запити від клієнта, перевіряє їх коректність, виконує бізнес-логіку, звертається до PostgreSQL через бібліотеку pg, за потреби формує запит до OpenAI API та повертає результат у форматі JSON. Серверні маршрути згруповано за модулями: автентифікація, профіль, вправи, тренувальні плани, історія тренувань, nutrition API, Dashboard, рекомендації та AI-узагальнення. Авторизація реалізована на основі JWT: після успішного входу сервер формує токен, який клієнт додає до захищених запитів, а middleware перевіряє його коректність. Паролі захищено хешуванням bcrypt [14; 15].

Обмін даними здійснюється через REST API з використанням стандартних HTTP-методів: GET для отримання даних, POST для створення, PUT/PATCH для оновлення та DELETE для видалення [19; 20; 21; 22]. Основними модулями вебплатформи є модуль автентифікації, профілю, вправ, тренувальних планів, історії тренувань, харчування, Dashboard, рекомендаційний модуль і модуль AI-узагальнення. Dashboard виконує роль центрального інформаційного екрана, що об’єднує профіль, активний план, прогрес виконання, підсумок харчування за день і рекомендації. OpenAI API не приймає основних рішень і не є власною нейронною мережею системи – він лише формує зрозумілий текстовий підсумок підготовлених сервером рекомендацій.

Загальну клієнт-серверну архітектуру вебплатформи з розподілом на рівні наведено на рисунку 2.1.

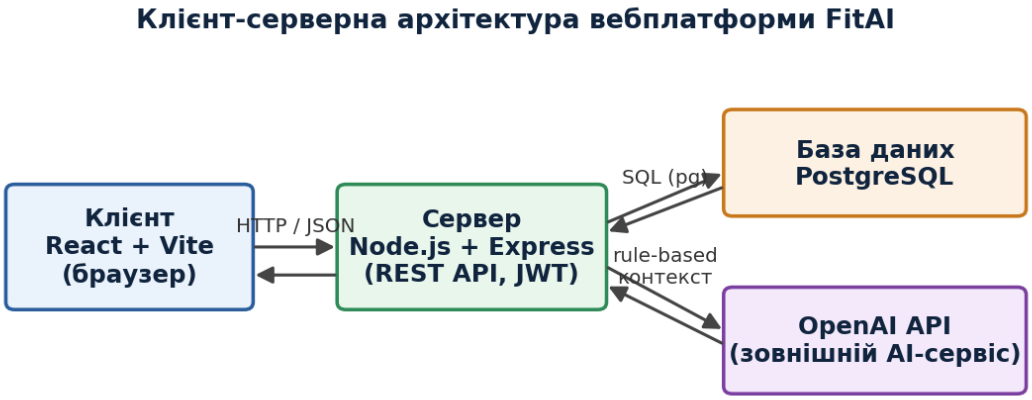

Рисунок 2.1 – Клієнт-серверна архітектура вебплатформи FitAI

Загальна схема взаємодії компонентів така: користувач працює з інтерфейсом React у браузері; клієнтська частина надсилає HTTP-запити до REST API; сервер Express обробляє їх, перевіряє JWT-токен, виконує бізнес-логіку та звертається до PostgreSQL; база даних повертає записи; сервер формує rule-based рекомендації; за потреби передає структуровані дані до OpenAI API для текстового узагальнення; після цього backend повертає відповідь у форматі JSON, а frontend оновлює відповідні елементи інтерфейсу. Для передавання даних використовується формат JSON, що безпосередньо підтримується JavaScript і легко обробляється на обох сторонах.

Обрана архітектура також створює основу для подальшого розвитку: до системи можуть бути додані розширена аналітика прогресу, детальніша статистика харчування, ширша база продуктів або складніші алгоритми персоналізації. Розподіл відповідальності між клієнтом, сервером і базою даних спрощує підтримку, дає змогу окремо розвивати інтерфейс і серверну логіку та забезпечує безпечну роботу з персональними даними користувача, оскільки всі операції з даними проходять через серверний рівень.

2.3 Проєктування бази даних PostgreSQL та ER-моделі

Базу даних FitAI реалізовано на реляційній СУБД PostgreSQL. Такий вибір обґрунтований тим, що предметна область має чітко виражені

структуровані сутності та зв’язки: один користувач має профіль, може мати декілька тренувальних планів, кожен план містить тренування по днях, тренування складаються з вправ, а харчові записи належать конкретному користувачу та даті. Реляційна модель дає змогу формалізувати ці зв’язки за допомогою первинних і зовнішніх ключів та забезпечити цілісність даних [9; 10; 11; 12; 13].

У моделі бази даних передбачено такі основні таблиці: users, user_profiles, exercises, workout_plans, workouts, workout_exercises, workout_logs, nutrition_targets, nutrition_logs, recommendations. Узагальнену характеристику таблиць наведено в таблиці 2.3.

Таблиця 2.3 – Основні таблиці бази даних FitAI

Таблиця	**Призначення**	**Ключові зв’язки**
users	Облікові записи користувачів (email, хеш пароля, ім’я)	Базова для всіх персональних даних
user_profiles	Персональні параметри (вік, стать, зріст, вага, рівень, ціль)	1:1 з users
exercises	База вправ (назва, опис, група м’язів, складність)	Використовується у workout_exercises
workout_plans	Персональні тренувальні плани	1:N від users
workouts	Окремі тренування (дні) у межах плану	1:N від workout_plans
workout_exercises	Склад тренування (підходи, повторення, порядок)	M:N між workouts та exercises
workout_logs	Історія виконаних тренувань	N:1 до users і workouts
nutrition_targets	Добові цілі харчування (калорії, БЖВ, вода)	1:1/1:N з users
nutrition_logs	Записи журналу харчування за дату	1:N від users
recommendations	Сформовані рекомендації користувачу	1:N від users

Центральною сутністю ER-моделі є User. З нею пов’язані UserProfile (зв’язок «один до одного»), WorkoutPlan (1:N), WorkoutLog (1:N), NutritionTarget, NutritionLog (1:N) та Recommendation (1:N). Тренувальний план деталізується через Workout (дні плану, 1:N), а зв’язок тренувань із вправами реалізовано через проміжну сутність WorkoutExercise, що забезпечує відношення «багато до багатьох» між Workout та Exercise і зберігає параметри виконання (підходи, повторення, порядок). Така нормалізована структура усуває дублювання: опис вправи зберігається один раз, облікові дані

відокремлені від профільних параметрів, а добові цілі – від фактичних записів журналу.

Основними зв’язками бази даних FitAI є:

– users → user_profiles: один користувач має один профіль (1:1);

– users → workout_plans: один користувач може мати декілька тренувальних планів (1:N);

– workout_plans → workouts: один план складається з декількох тренувальних днів (1:N);

– workouts → workout_exercises ← exercises: склад дня описується через проміжну таблицю (M:N);

– users → workout_logs ← workouts: історія виконання пов’язана з користувачем і тренуванням (1:N);

– users → nutrition_targets: набір добових цілей харчування користувача;

– users → nutrition_logs: записи журналу харчування за різні дати (1:N);

– users → recommendations: сформовані для користувача рекомендації (1:N).

Така структура дозволяє уникнути дублювання даних: інформація про вправу зберігається один раз у таблиці exercises, а її використання у конкретному тренуванні описується через workout_exercises, тому одна вправа може входити до різних планів без повторного збереження опису. Аналогічно облікові дані відокремлені від профільних параметрів, тренувальні плани – від історії виконання, а добові цілі харчування – від фактичних записів журналу. Кожен функціональний модуль працює з відповідними таблицями: автентифікація – з users, профіль – з user_profiles, генерація плану створює записи у workout_plans, workouts і workout_exercises, історія використовує workout_logs, а модуль харчування – nutrition_targets і nutrition_logs.

ER-модель FitAI побудована з урахуванням нормалізації даних. Облікові дані користувача, профіль, довідник вправ, тренувальні плани, дні тренувань, склад тренувальних днів, історія виконання, добові цілі харчування, записи журналу та рекомендації зберігаються окремо. Це зменшує дублювання

інформації, спрощує оновлення вправ, забезпечує гнучке формування різних планів, дозволяє вести харчові записи за різні дати та використовувати накопичені дані для формування рекомендацій. Для окремих полів (рівень підготовки, ціль, статус плану, оцінка складності, числові показники харчування) доцільно застосовувати обмеження допустимих значень, що зменшує ризик появи некоректних даних.

Узагальнену ER-модель бази даних із основними сутностями та зв’язками наведено на рисунку 2.2.

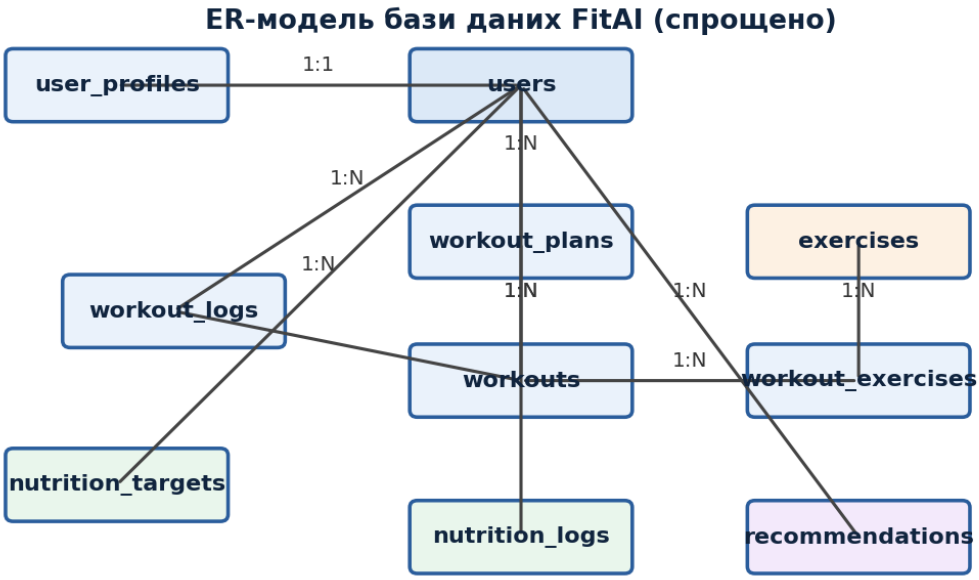

Рисунок 2.2 – ER-модель бази даних FitAI

Для підтримання цілісності даних використовуються первинні та зовнішні ключі: user_id у таблицях профілю, планів, історії та харчування посилається на users; plan_id у workouts – на workout_plans; workout_id та exercise_id у workout_exercises – на відповідні таблиці. Це унеможливлює появу «осиротілих» записів. Для прискорення типових запитів доцільно індексувати поля email у users, user_id у залежних таблицях, дату виконання тренування у workout_logs та дату запису в nutrition_logs. Підсумок

харчування за день формується агрегуванням записів nutrition_logs і порівнюється з nutrition_targets, тому не дублюється у базі даних. Інтеграція з OpenAI API не потребує окремих таблиць для зберігання параметрів моделі чи навчальних даних.

2.4 Проєктування REST API та сценаріїв взаємодії

Взаємодію клієнта й сервера побудовано на REST API [19; 20]. Клієнтська частина надсилає HTTP-запити, серверна – приймає їх, перевіряє вхідні дані, виконує логіку, звертається до PostgreSQL, за потреби взаємодіє з OpenAI API та повертає відповідь у форматі JSON [21; 22]. Публічними є лише маршрути реєстрації та авторизації; решта потребує JWT-авторизації, оскільки працює з персональними даними. Після входу сервер формує JWT-токен, який клієнт передає в заголовку Authorization у форматі Bearer token [14; 15]; middleware перевіряє токен і визначає користувача, обмежуючи доступ лише його даними [23]. Основні endpoint-и наведено в таблиці 2.4.

Таблиця 2.4 – Основні endpoint-и REST API вебплатформи FitAI

Метод	Endpoint	Призначення	Авт.
POST	/api/auth/register	Реєстрація нового користувача	Ні
POST	/api/auth/login	Авторизація, видача JWT-токена	Ні
GET	/api/profile	Отримання профілю користувача	Так
POST	/api/profile	Створення профілю користувача	Так
PUT	/api/profile	Оновлення профілю користувача	Так
GET	/api/exercises	Список вправ із бази даних	Так
GET	/api/exercises/:id	Детальна інформація про вправу	Так
GET	/api/exercises?...	Фільтрація вправ за параметрами	Так
POST	/api/workout-plans/generate	Генерація персонального плану	Так
GET	/api/workout-plans/active	Активний тренувальний план	Так
GET	/api/workout-plans/:id	Перегляд конкретного плану	Так
GET	/api/workout-plans/:id/days	Перегляд плану по днях	Так
POST	/api/workouts/:id/complete	Позначення тренування виконаним	Так
GET	/api/workout-history	Історія виконаних тренувань	Так

GET	/api/nutrition/targets	Отримання добових цілей харчування	Так
POST	/api/nutrition/targets	Створення добових цілей	Так
PUT	/api/nutrition/targets	Оновлення добових цілей	Так
GET	/api/nutrition/logs?date=	Записи журналу харчування за дату	Так
POST	/api/nutrition/logs	Додавання запису журналу	Так
PUT	/api/nutrition/logs/:id	Оновлення запису журналу	Так
DELETE	/api/nutrition/logs/:id	Видалення запису журналу	Так
GET	/api/nutrition/summary?date=	Підсумок харчування за день	Так
GET	/api/recommendations	Рекомендації на основі правил	Так
POST	/api/recommendations/ai-summary	AI-узагальнення рекомендацій	Так
GET	/api/dashboard	Зведена інформація для Dashboard	Так

Узагальнену послідовність обробки авторизованого запиту – від клієнта через сервер до бази даних і зовнішнього AI-сервісу – наведено на рисунку 2.3.

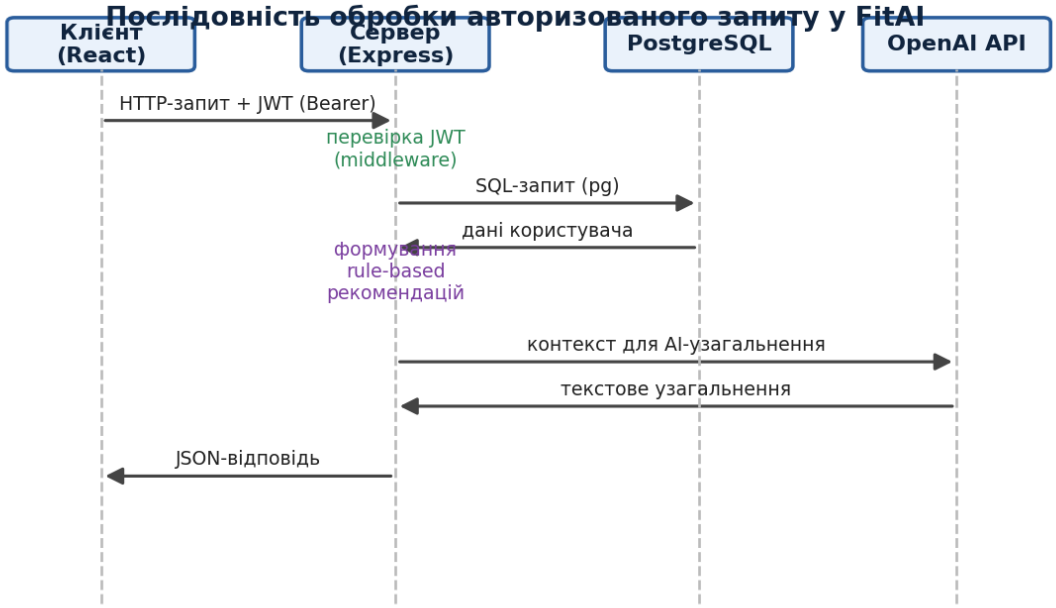

Рисунок 2.3 – Послідовність обробки авторизованого запиту у FitAI

Розглянемо призначення ключових endpoint-ів детальніше. Маршрут /api/auth/register приймає email і пароль, перевіряє унікальність email, хешує пароль засобами bcrypt і створює запис користувача; пароль у відкритому вигляді не зберігається. Маршрут /api/auth/login перевіряє облікові дані та повертає JWT-токен. Маршрути /api/profile забезпечують отримання, створення й оновлення профілю, дані якого використовуються для

персоналізації. База вправ доступна через /api/exercises із фільтрацією через query-параметри (наприклад, /api/exercises?difficulty=beginner&muscleGroup=legs), що дає змогу не створювати окремий маршрут для кожного варіанта пошуку.

Генерація плану виконується через /api/workout-plans/generate, перегляд – через /api/workout-plans/active та /api/workout-plans/:id/days. Виконання тренування фіксується через /api/workouts/:id/complete, а історія доступна через /api/workout-history. Окрему групу становить nutrition API: /api/nutrition/targets для добових цілей, /api/nutrition/logs для журналу та /api/nutrition/summary для підсумку за день, який формується агрегуванням записів і порівнянням із цілями. Маршрут /api/recommendations повертає rule-based рекомендації, /api/recommendations/ai-summary – їх текстове AI-узагальнення, а /api/dashboard агрегує дані головної панелі одним запитом, зменшуючи кількість окремих звернень із клієнтської частини.

Основні сценарії взаємодії охоплюють повний цикл роботи користувача. Реєстрація створює обліковий запис із хешуванням пароля; авторизація повертає JWT-токен. Робота з профілем передбачає отримання, створення й оновлення параметрів, які використовуються для персоналізації. Перегляд вправ підтримує фільтрацію через query-параметри. Генерація плану ініціює серверну логіку підбору вправ на основі профілю та зберігає результат у workout_plans, workouts і workout_exercises. Виконання тренування створює запис у workout_logs, що впливає на історію, Dashboard і рекомендації. Робота з модулем харчування охоплює добові цілі, додавання записів журналу та формування підсумку за день агрегуванням записів. Окремий dashboard endpoint повертає зведену інформацію одним запитом. У разі недоступності OpenAI API основні функції залишаються доступними, а користувач бачить базові rule-based рекомендації.

2.5 Логіка рекомендаційного модуля

Рекомендаційний модуль забезпечує персоналізовану інформаційну підтримку користувача на основі даних системи. Його реалізовано як

пояснювану експертну систему на основі правил: система не навчається самостійно, не використовує власну нейронну мережу та не виконує машинного навчання [26; 27]. Рекомендації формуються за заздалегідь визначеними умовами, що описують типові ситуації роботи користувача. Такий підхід є контрольованим, передбачуваним і пояснюваним: для кожної рекомендації можна визначити, які саме дані вплинули на її появу – відсутність профілю чи його параметрів, відсутність активного плану, низька регулярність виконання, порожній журнал харчування або відхилення фактичних показників від цілей.

Модуль працює з кількома групами даних: параметри профілю; стан тренувального плану; історія виконання тренувань; дані модуля харчування (цілі, журнал, підсумок за день); службові дані для Dashboard та AI-узагальнення. Логіка реалізується на серверній стороні: сервер отримує дані поточного користувача з PostgreSQL, послідовно застосовує набір правил і формує структурований список рекомендацій, кожна з яких має тип, заголовок, текст, причину та рівень важливості. Пріоритет визначає, які рекомендації показати першими (наприклад, незаповнений профіль важливіший за незначне відхилення харчових показників). Основні правила наведено в таблиці 2.5.

Таблиця 2.5 – Правила рекомендаційного модуля FitAI

Код	Група	Умова	Рекомендація
REC-01	Профіль	Профіль відсутній	Заповнити профіль для персоналізації
REC-02	Профіль	Відсутні ключові параметри	Доповнити профіль основними параметрами
REC-03	Профіль	Рівень підготовки не вказано	Вказати рівень для підбору вправ
REC-04	Профіль	Ціль тренувань не вказано	Визначити ціль тренувань
REC-05	План	Активний план відсутній	Згенерувати персональний план
REC-06	План	План без тренувальних днів	Перегенерувати або перевірити план
REC-07	План	Початковий рівень, складний план	Переглянути складність плану
REC-08	План	План не розпочато	Почати виконання з першого дня

REC-09	Історія	Немає виконаних тренувань	Позначати тренування виконаними
REC-10	Історія	Виконано частину тренувань	Підтримувати регулярність
REC-11	Історія	Давно не було тренувань	Поступово відновити регулярність
REC-12	Історія	Оцінка складності занадто висока	Обрати простіший план
REC-13	Історія	Оцінка складності стабільно низька	Поступово підвищити складність
REC-14	Харчування	Добові цілі відсутні	Заповнити добові цілі харчування
REC-15	Харчування	Журнал за день порожній	Додати записи харчування
REC-16	Харчування	Калорії значно нижчі за ціль	Перевірити повноту записів
REC-17	Харчування	Калорії значно вищі за ціль	Звернути увагу на перевищення цілі
REC-18	Харчування	Вода нижча за ціль	Звернути увагу на показник води
REC-19	Харчування	Внесені калорії без БЖВ	Уточнити білки, жири, вуглеводи
REC-20	Харчування	БЖВ відрізняються від цілей	Перевірити співвідношення макронутрієнтів
REC-21	Dashboard	Бракує ключових даних	Заповнити профіль і створити план
REC-22	Dashboard	Немає активності за день	Почати план або додати записи
REC-23	Система	Недостатньо даних для персоналізації	Додати більше інформації
REC-24	AI-узагальнення	Сформовано кілька рекомендацій	Згенерувати текстове узагальнення
REC-25	AI-узагальнення	OpenAI API недоступний	Показати базові рекомендації без AI

Загальний алгоритм роботи рекомендаційного модуля можна подати як послідовність кроків: користувач авторизується; клієнтська частина надсилає запит на отримання рекомендацій; сервер перевіряє JWT-токен і визначає користувача; отримує з бази даних профіль, активний план, історію виконання, добові цілі та записи журналу харчування за поточну дату; обчислює підсумок харчування за день; застосовує набір правил; формує структурований список рекомендацій; за потреби передає їх до сервісу AI-узагальнення; повертає клієнтові базові рекомендації та, якщо доступно, текстове AI-узагальнення.

Правила згруповано за блоками. Блок профілю перевіряє повноту персональних даних і пропонує завершити їх заповнення. Блок тренувального

плану рекомендує створити план за його відсутності або переглянути складність, якщо вона не відповідає рівню користувача. Блок історії аналізує регулярність виконання та може нагадати про необхідність позначати тренування виконаними або, навпаки, повідомити про прогрес. Блок харчування працює з добовими цілями та журналом: пропонує заповнити цілі, додати записи або звертає увагу на відхилення фактичних показників від цільових. Блок Dashboard формує короткі першочергові підказки для головної сторінки. Рекомендації щодо харчування формулюються нейтрально («зверніть увагу», «перевірте заповнення журналу») і не є директивними медичними вказівками.

Інтеграція з OpenAI API використовується не для заміни основного модуля, а для генерації текстового узагальнення вже сформованих рекомендацій [24; 25]. Сервер готує обмежений структурований контекст (узагальнені показники, а не сирі персональні дані), звертається до зовнішнього сервісу та повертає текстовий підсумок. Ключ OpenAI API зберігається лише на серверній стороні у змінних середовища і не передається клієнту. Якщо сервіс недоступний, модуль продовжує працювати, показуючи базові rule-based рекомендації. Такий підхід узгоджує фактичний функціонал FitAI із темою роботи, не завищуючи рівень реалізованої AI-функціональності.

Серверну реалізацію рекомендаційного модуля доцільно будувати у вигляді окремого сервісу (recommendationService), що отримує з бази даних потрібні дані, застосовує правила та повертає масив рекомендацій, тоді як контролер відповідає за обробку HTTP-запиту й повернення відповіді. Для AI-узагальнення окремий сервіс (aiSummaryService) готує запит до OpenAI API та обробляє відповідь. Для користувача важливо бачити не лише саме повідомлення, а й коротке пояснення причини; наприклад, замість абстрактного «покращіть регулярність» система показує: «у вас є активний план, але ще немає виконаних тренувань; позначайте тренування як виконані,

щоб відстежувати прогрес». Такий формат відповідає ідеї пояснюваної експертної системи.

Висновки до розділу 2

У другому розділі було виконано проєктування вебплатформи FitAI з урахуванням визначених потреб користувачів і фактичного функціонального призначення системи. Сформульовано функціональні та нефункціональні вимоги, які охоплюють реєстрацію, авторизацію, роботу з профілем, базою вправ, тренувальними планами, історією виконаних тренувань, модулем харчування, Dashboard, рекомендаціями та AI-узагальненням.

Було обґрунтовано загальну архітектуру FitAI як клієнт-серверної вебплатформи. Клієнтська частина реалізується на React і Vite, серверна частина – на Node.js та Express, а для збереження даних використовується PostgreSQL. Взаємодія між клієнтською та серверною частинами здійснюється через REST API, а доступ до приватних функцій системи захищається за допомогою JWT.

Окрему увагу приділено проєктуванню бази даних. Визначено основні сутності системи: користувачі, профілі, вправи, тренувальні плани, тренувальні дні, історія виконання тренувань, добові цілі харчування, записи журналу харчування та рекомендації. Така структура забезпечує логічний зв’язок між основними модулями FitAI та дає змогу зберігати персональні дані користувача між сеансами роботи.

У межах проєктування REST API було визначено основні групи endpoint-ів, які забезпечують роботу модулів автентифікації, профілю, вправ, тренувальних планів, історії, харчування, Dashboard, рекомендацій і AI-узагальнення. Це дозволяє відокремити клієнтську логіку від серверної та забезпечити централізовану обробку запитів.

Також було спроєктовано рекомендаційний модуль FitAI. Його основна логіка ґрунтується на правилах, які аналізують профіль користувача, активний тренувальний план, історію виконаних тренувань, добові цілі харчування та записи журналу. OpenAI API розглядається як зовнішній сервіс для текстового

узагальнення вже сформованих рекомендацій, а не як власна нейронна мережа FitAI або самостійний механізм прийняття рішень.

Отже, у другому розділі було сформовано проєктну основу вебплатформи FitAI: визначено вимоги, архітектуру, структуру бази даних, API та принцип роботи рекомендаційного модуля. Ці рішення стали підґрунтям для подальшої реалізації, тестування та демонстрації програмного продукту.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБПЛАТФОРМИ FITAI

3.1 Реалізація серверної частини Node.js / Express

Серверна частина вебплатформи FitAI реалізована з використанням Node.js та Express [9; 12]. Вона виконує роль прикладного рівня системи, який приймає HTTP-запити від клієнтської частини, обробляє їх, перевіряє авторизацію користувача, взаємодіє з базою даних PostgreSQL і повертає результат у форматі JSON.

Backend забезпечує роботу основних функціональних модулів FitAI: реєстрації та авторизації користувача, профілю, бази вправ, генерації тренувального плану, історії виконаних тренувань, модуля харчування, Dashboard, рекомендаційного модуля та AI-узагальнення через зовнішній сервіс OpenAI API [24; 25].

Express використано для побудови REST API та маршрутизації запитів. Серверна логіка поділена за функціональними напрямами: окремі маршрути відповідають за автентифікацію, профіль, вправи, тренувальні плани, історію тренувань, харчування, Dashboard і рекомендації. Такий поділ спрощує підтримку коду та дає змогу розширювати систему без повної зміни її структури.

Основними елементами серверної частини є маршрути, контролери, middleware та сервіси. Маршрути визначають адреси API, до яких звертається клієнтська частина. Контролери обробляють конкретні запити та формують відповіді. Middleware використовується для проміжної перевірки запитів, зокрема для перевірки JWT-токена. Сервіси містять допоміжну бізнес-логіку, яку недоцільно розміщувати безпосередньо в контролерах.

Модуль автентифікації реалізує реєстрацію та вхід користувача. Під час реєстрації сервер отримує дані користувача, перевіряє їх і зберігає новий обліковий запис у PostgreSQL. Пароль перед збереженням хешується за допомогою bcrypt, тому в базі даних не зберігається відкритий пароль [16]. Під

час входу сервер перевіряє email і пароль користувача, після чого формує JWT-токен для подальшого доступу до захищених функцій системи.

JWT використовується для захисту приватних маршрутів. Після авторизації клієнтська частина додає токен до запитів до backend [14; 15]. Серверний middleware перевіряє наявність і коректність токена, визначає користувача та передає його ідентифікатор до наступних обробників. Якщо токен відсутній або некоректний, сервер не надає доступ до персональних даних користувача.

Робота з PostgreSQL реалізована виключно на серверному рівні. Клієнтська частина не звертається до бази даних напряму. Усі операції з профілем, вправами, тренувальними планами, історією, харчовими цілями, записами журналу та рекомендаціями проходять через REST API. Це дозволяє централізовано контролювати доступ до даних і забезпечувати, щоб користувач отримував лише власну інформацію.

Для роботи з профілем користувача backend реалізує отримання, створення та оновлення персональних параметрів. Дані профілю використовуються під час генерації тренувального плану та формування рекомендацій. Якщо профіль відсутній або заповнений не повністю, система може повідомити користувача про необхідність його заповнення.

Модуль вправ забезпечує отримання переліку вправ із бази даних і підтримує їх фільтрацію за передбаченими параметрами. Дані вправ використовуються як для перегляду користувачем, так і для формування тренувального плану. Під час генерації плану backend враховує профіль користувача та доступні вправи, після чого створює записи тренувального плану в базі даних.

Модуль історії тренувань реалізує фіксацію виконаних тренувань. Коли користувач позначає тренування як виконане, клієнтська частина надсилає відповідний запит до backend, а сервер створює запис в історії. Ці дані надалі використовуються для відображення прогресу на Dashboard і для формування рекомендацій щодо регулярності тренувань.

Модуль харчування реалізує роботу з добовими цілями та журналом харчування. Backend забезпечує створення або оновлення добових цілей, додавання записів до журналу, отримання записів за обрану дату та формування підсумку за день. До основних показників належать калорії, білки, жири, вуглеводи та вода. Дані модуля харчування використовуються на NutritionPage, Dashboard і в рекомендаційному модулі.

Dashboard формується серверною частиною як зведення ключових даних користувача. Для цього backend може отримувати інформацію з профілю, активного тренувального плану, історії виконаних тренувань, добових цілей харчування, журналу харчування та рекомендацій. Такий підхід дозволяє клієнтській частині відображати узагальнений стан користувача в одному місці.

Рекомендаційний модуль реалізовано на серверній стороні як систему правил. Backend аналізує профіль користувача, активний тренувальний план, історію виконання тренувань, добові цілі харчування та записи журналу. На основі цих даних система формує структуровані рекомендації. Такий підхід є пояснюваним, оскільки кожна рекомендація пов’язана з конкретною умовою, яка спрацювала під час аналізу даних.

Інтеграція з OpenAI API реалізована як допоміжна серверна функція. Спочатку FitAI формує рекомендації за внутрішніми правилами, а вже після цього backend може передати підготовлений контекст до OpenAI API для створення короткого текстового узагальнення. OpenAI API не замінює основну рекомендаційну логіку системи і не є власною нейронною мережею FitAI.

Налаштування backend зберігаються у змінних середовища. До таких параметрів належать дані підключення до PostgreSQL, секрет JWT, порт запуску серверної частини та ключ OpenAI API. Використання змінних середовища дозволяє не розміщувати службові й секретні значення безпосередньо в коді та відокремити конфігурацію від програмної логіки.

Під час реалізації серверної частини передбачено обробку типових помилок. До таких ситуацій належать некоректні облікові дані, відсутність авторизації, незаповнений профіль, відсутність активного тренувального плану, помилки під час роботи з базою даних або недоступність зовнішнього AI-сервісу. У разі помилки OpenAI API базові рекомендації, сформовані системою правил, мають залишатися доступними користувачу.

Отже, серверна частина FitAI об’єднує основну прикладну логіку вебплатформи: автентифікацію, захист приватних маршрутів, роботу з PostgreSQL, тренувальне планування, модуль харчування, Dashboard, рекомендації та AI-узагальнення. Саме backend забезпечує зв’язок між клієнтським інтерфейсом React, базою даних PostgreSQL і зовнішнім сервісом OpenAI API.

3.2 Реалізація клієнтської частини React

Клієнтська частина вебплатформи FitAI реалізована з використанням React та Vite [1; 2; 3; 4; 5]. Вона відповідає за побудову користувацького інтерфейсу, відображення сторінок, обробку дій користувача та взаємодію із серверною частиною через REST API. Саме через клієнтську частину користувач працює з основними функціями системи: реєстрацією, авторизацією, профілем, базою вправ, тренувальним планом, історією виконаних тренувань, модулем харчування, Dashboard і рекомендаціями.

React використано для побудови інтерфейсу за компонентним принципом. Це дає змогу поділити клієнтську частину на окремі сторінки, форми, навігаційні елементи, картки вправ, блоки тренувального плану, елементи журналу харчування, інформаційні блоки Dashboard і компоненти рекомендацій. Такий підхід спрощує супровід коду, повторне використання елементів інтерфейсу та подальше розширення системи.

Vite використовується як інструмент запуску та збирання клієнтської частини. Він забезпечує швидкий запуск середовища розроблення, зручну

роботу з React-компонентами та формування фінальної збірки вебзастосунку для подальшого розгортання.

Клієнтська частина FitAI побудована як односторінковий вебзастосунок. Переходи між основними сторінками відбуваються без повного перезавантаження браузера. Для маршрутизації використовується React Router. Він забезпечує переходи між сторінками реєстрації, авторизації, Dashboard, профілю користувача, бази вправ, тренувального плану, історії тренувань, сторінки харчування NutritionPage і сторінки або блоку рекомендацій.

Основними сторінками клієнтської частини FitAI є: 1. сторінка реєстрації користувача; 2. сторінка авторизації користувача; 3. Dashboard; 4. сторінка профілю користувача; 5. сторінка бази вправ; 6. сторінка тренувального плану; 7. сторінка історії виконаних тренувань; 8. сторінка харчування NutritionPage; 9. сторінка або блок рекомендацій; 10. блок текстового AI-узагальнення рекомендацій.

Сторінка реєстрації призначена для створення нового облікового запису користувача. Після введення необхідних даних клієнтська частина надсилає POST-запит до серверного API. У разі успішної реєстрації користувач може перейти до сторінки авторизації.

Сторінка авторизації забезпечує вхід користувача до системи. Користувач вводить email і пароль, після чого клієнтська частина надсилає відповідний запит до backend. Якщо облікові дані коректні, сервер повертає JWT-токен. Надалі цей токен використовується для доступу до приватних сторінок і захищених endpoint-ів API.

Для захисту приватних сторінок у клієнтській частині перевіряється наявність токена авторизації. Якщо користувач не авторизований, доступ до Dashboard, профілю, тренувального плану, історії тренувань, NutritionPage і рекомендацій має бути обмежений. Така перевірка на рівні інтерфейсу доповнює серверну перевірку JWT-токена, але не замінює її.

Сторінка профілю користувача дає змогу створювати або оновлювати персональні параметри, які використовуються для персоналізації тренувального плану та рекомендацій. До таких параметрів належать дані, передбачені реалізованою моделлю профілю, зокрема рівень фізичної підготовки та тренувальна ціль. Після збереження змін клієнтська частина надсилає оновлені дані до backend, а сервер записує їх у PostgreSQL.

Сторінка бази вправ відображає перелік вправ, отриманий із серверної частини. Користувач може переглядати вправи та застосовувати фільтрацію за передбаченими параметрами. Дані вправ використовуються не лише для ознайомлення, а й для формування персонального тренувального плану.

Сторінка тренувального плану призначена для перегляду активного плану користувача. На ній відображаються тренувальні дні та вправи, що входять до кожного дня. Користувач може позначити тренування як виконане. Після цього клієнтська частина надсилає відповідний запит до backend, а сервер створює запис в історії виконаних тренувань.

Сторінка історії тренувань відображає виконані користувачем тренування. Вона дає змогу переглянути, які тренування вже були завершені, і використовується як один із засобів контролю регулярності. Дані історії також враховуються на Dashboard і в рекомендаційному модулі.

NutritionPage реалізує інтерфейс модуля харчування. На цій сторінці користувач може створювати або редагувати добові цілі харчування, додавати записи до журналу та переглядати підсумок за день. До основних показників належать калорії, білки, жири, вуглеводи та вода. Після додавання або зміни даних клієнтська частина звертається до nutrition API серверної частини, а отримані результати відображаються в інтерфейсі.

Dashboard є центральною сторінкою користувача після авторизації. Він об’єднує дані з кількох модулів: профіль користувача, активний тренувальний план, прогрес виконання тренувань, короткий підсумок харчування та

рекомендації. Завдяки цьому користувач може швидко оцінити поточний стан роботи з вебплатформою без переходу між усіма сторінками.

Взаємодія клієнтської частини із серверною реалізована через HTTP-запити до REST API. Для цього використовується Axios. Під час запитів до захищених endpoint-ів JWT-токен додається до заголовків запиту. Це дозволяє серверу визначити авторизованого користувача та повернути лише ті дані, які належать йому.

У клієнтській частині передбачено обробку основних станів інтерфейсу: завантаження даних, успішне отримання відповіді, відсутність потрібної інформації та помилка запиту. Наприклад, якщо користувач ще не має активного тренувального плану, інтерфейс може запропонувати створити його. Якщо журнал харчування за день порожній, система може показати відповідне повідомлення. Якщо AI-узагальнення тимчасово недоступне, базові рекомендації мають залишатися доступними.

Окрему увагу приділено тому, що клієнтська частина не повинна містити секретних значень. Зокрема, ключ OpenAI API не зберігається у React-коді, оскільки клієнтська частина виконується в браузері користувача. Усі звернення до зовнішнього AI-сервісу здійснюються через backend.

У результаті реалізації клієнтської частини створено інтерфейс, який забезпечує виконання основних сценаріїв роботи FitAI: реєстрації, авторизації, заповнення профілю, перегляду вправ, генерації та перегляду тренувального плану, фіксації виконаних тренувань, перегляду історії, ведення журналу харчування, роботи з Dashboard і перегляду рекомендацій. Клієнтська частина взаємодіє із сервером через REST API та забезпечує користувачу доступ до функцій вебплатформи у зрозумілому форматі.

3.3 Реалізація роботи з PostgreSQL

Для збереження даних вебплатформи FitAI використано реляційну систему управління базами даних PostgreSQL [9; 10; 11; 12]. База даних є центральним сховищем інформації про користувачів, профілі, вправи,

тренувальні плани, історію виконаних тренувань, добові цілі харчування, записи журналу харчування та рекомендації.

Взаємодія з PostgreSQL здійснюється на серверній стороні. Клієнтська частина React не має прямого доступу до бази даних, а всі операції виконуються через REST API серверної частини Node.js / Express. Такий підхід дає змогу централізовано контролювати доступ до даних, перевіряти авторизацію користувача та обмежувати доступ лише до тих записів, які належать поточному користувачу.

Підключення до PostgreSQL реалізовано у серверній частині через окремий модуль роботи з базою даних. Параметри підключення зберігаються у змінних середовища, що дозволяє не фіксувати адресу бази даних, ім’я користувача, пароль та інші службові параметри безпосередньо в коді. Це спрощує перенесення системи між локальним середовищем розроблення, демонстраційним середовищем і потенційним серверним розгортанням.

У структурі бази даних FitAI використовуються пов’язані таблиці, що відповідають основним сутностям системи. Таблиця користувачів зберігає облікові записи, необхідні для реєстрації та авторизації. Таблиця профілів містить персональні параметри користувача, які використовуються для персоналізації тренувального плану та рекомендацій. Таблиця вправ містить перелік вправ і їхні характеристики. Таблиці тренувальних планів і тренувальних днів зберігають структуру персонального плану. Таблиця історії тренувань фіксує виконані користувачем тренування. Таблиці модуля харчування зберігають добові цілі та фактичні записи журналу харчування. Окремі записи рекомендацій можуть використовуватися для збереження або відображення сформованих системою підказок.

Основні таблиці бази даних FitAI можна згрупувати за функціональними модулями системи. Модуль автентифікації працює з даними користувачів. Модуль профілю працює з персональними параметрами користувача. Модуль вправ використовує таблицю вправ. Модуль тренувального планування працює з таблицями планів, тренувальних днів і вправ у межах плану. Модуль

історії використовує записи про виконані тренування. Модуль харчування працює з добовими цілями та журналом харчування. Dashboard отримує узагальнені дані з кількох таблиць. Рекомендаційний модуль аналізує профіль, активний план, історію виконання тренувань і харчові показники.

Під час реєстрації нового користувача серверна частина створює запис у таблиці користувачів. Перед збереженням пароль хешується, тому у базі даних не зберігається відкритий пароль. Під час авторизації backend отримує облікові дані, знаходить відповідного користувача в PostgreSQL, перевіряє пароль і формує JWT-токен для подальшої роботи із захищеними функціями системи.

Після авторизації користувач може створити або оновити профіль. Дані профілю зберігаються в базі даних і пов’язуються з конкретним користувачем. Профіль є важливим елементом персоналізації, оскільки саме він містить параметри, які враховуються під час генерації тренувального плану та формування рекомендацій.

База вправ зберігається в PostgreSQL як структурований набір записів. Кожна вправа може містити назву, опис і параметри, які використовуються для фільтрації та підбору вправ у межах тренувального плану. Під час відкриття сторінки вправ клієнтська частина надсилає запит до backend, сервер отримує відповідні записи з бази даних і повертає їх у форматі JSON.

Генерація тренувального плану пов’язана з кількома таблицями. Спочатку сервер отримує профіль користувача та доступні вправи, після чого формує структуру плану. Згенерований план зберігається в базі даних як набір пов’язаних записів: загальний тренувальний план, окремі тренувальні дні та вправи, що входять до цих днів. Завдяки цьому користувач може повернутися до плану після повторного входу до системи.

Історія виконаних тренувань також зберігається у PostgreSQL. Коли користувач позначає тренування як виконане, backend створює відповідний запис, пов’язаний із користувачем і тренуванням. Ця інформація використовується для відображення прогресу, перегляду історії та

формування рекомендацій. Наявність історії дає змогу системі не обмежуватися статичним планом, а враховувати фактичні дії користувача.

Модуль харчування використовує окремі таблиці для добових цілей і журналу харчування. Добові цілі містять заплановані значення калорій, білків, жирів, вуглеводів і води. Журнал харчування містить фактичні записи користувача за конкретну дату. На основі цих записів backend формує підсумок за день і може порівнювати фактичні значення з установленими цілями.

Dashboard отримує з PostgreSQL узагальнені дані з кількох модулів. Для його формування серверна частина може використовувати інформацію про профіль, активний тренувальний план, виконані тренування, харчові цілі, записи журналу харчування та рекомендації. Завдяки цьому Dashboard виконує роль центральної інформаційної сторінки, на якій користувач бачить основний стан роботи із системою.

Рекомендаційний модуль використовує дані, що вже зберігаються в PostgreSQL. Для формування рекомендацій backend аналізує профіль користувача, наявність активного тренувального плану, історію виконання тренувань, добові цілі харчування та фактичні записи журналу. Якщо певна умова виконується, система формує відповідну рекомендацію. Наприклад, якщо користувач не має активного плану, система може запропонувати його створити; якщо журнал харчування порожній, система може нагадати про необхідність внесення даних; якщо є виконані тренування, це може враховуватися для оцінки регулярності.

Важливим принципом роботи з базою даних є прив’язка персональних записів до конкретного користувача. Профіль, тренувальні плани, історія, харчові цілі, записи журналу та рекомендації мають бути пов’язані з ідентифікатором користувача. Під час виконання запитів backend використовує дані з JWT-токена для визначення поточного користувача та обмеження доступу до його власних записів.

Під час реалізації роботи з PostgreSQL враховано потребу в цілісності даних. Зв’язки між таблицями дають змогу логічно поєднати користувача з його профілем, планами, історією, харчуванням і рекомендаціями. Така структура зменшує ризик дублювання інформації та забезпечує узгоджене зберігання даних у межах різних модулів системи.

У разі помилок під час роботи з базою даних серверна частина має повертати клієнту коректне повідомлення про проблему. Наприклад, якщо профіль ще не створений, система не повинна аварійно завершувати роботу, а має повідомити користувача про необхідність заповнення профілю. Якщо активний тренувальний план відсутній, інтерфейс може запропонувати створити новий план.

Отже, PostgreSQL у FitAI використовується не лише як сховище окремих записів, а як основа для роботи всіх ключових модулів вебплатформи. Саме база даних забезпечує сталість інформації між сеансами роботи, зв’язок між профілем, планом, історією, харчуванням і рекомендаціями, а також створює основу для подальшого розширення функціональності системи.

3.4 Генерація тренувального плану та рекомендаційний модуль

Одним із центральних функціональних напрямів вебплатформи FitAI є реалізація персоналізованого тренувального планування. Цей модуль пов’язує профіль користувача, базу вправ, алгоритм формування плану, перегляд тренувань по днях, фіксацію виконання та подальше використання історії тренувань у рекомендаційному модулі.

Формування тренувального плану починається з перевірки наявності профілю користувача. Профіль містить параметри, необхідні для персоналізації: рівень підготовки, тренувальну ціль та інші дані, передбачені реалізованою моделлю. Якщо профіль відсутній або заповнений не повністю, система не повинна формувати некоректний план, а має запропонувати користувачу спочатку оновити профіль.

Після отримання даних профілю серверна частина звертається до бази вправ у PostgreSQL. Вправи зберігаються як структуровані записи з

параметрами, які можуть використовуватися для фільтрації та добору. На основі профілю користувача та доступної бази вправ backend формує персональний тренувальний план. Згенерований план зберігається у базі даних, що дає змогу користувачу переглядати його повторно після оновлення сторінки або нового входу до системи.

Тренувальний план відображається у клієнтській частині по днях. Такий формат є зручним для користувача, оскільки він бачить не лише загальний набір вправ, а й послідовність тренувань. У межах конкретного дня плану користувач може переглянути вправи, які потрібно виконати. Це робить план не статичним списком, а структурованим сценарієм тренувального процесу.

Після виконання тренування користувач може позначити його як виконане. Клієнтська частина надсилає відповідний запит до серверного API, а backend створює запис в історії тренувань. Історія виконання є важливою частиною системи, оскільки вона дозволяє відображати прогрес користувача та враховувати фактичну активність під час формування рекомендацій.

Другим важливим напрямом реалізації є модуль харчування. Він не розглядається як медична або дієтологічна система, а призначений для інформаційного супроводу користувача та фіксації введених ним даних. Модуль харчування дає змогу працювати з добовими цілями, вести журнал харчування та переглядати підсумок за день.

Добові цілі харчування містять орієнтовні показники калорій, білків, жирів, вуглеводів і води. Користувач може створити або змінити ці цілі через інтерфейс NutritionPage. Після збереження клієнтська частина надсилає дані до backend, а сервер записує їх у відповідну таблицю PostgreSQL. Надалі ці значення використовуються для порівняння з фактичними записами журналу.

Журнал харчування дає змогу користувачу додавати записи за конкретний день. Кожен запис містить харчові показники, які враховуються під час формування денного підсумку. Після додавання запису клієнтська частина оновлює відображення журналу та підсумкових значень. Backend

отримує дані журналу з PostgreSQL, агрегує їх за датою та повертає клієнтській частині результат у форматі JSON.

Підсумок харчування за день формується шляхом узагальнення записів журналу. Система підраховує фактичні значення калорій, білків, жирів, вуглеводів і води та порівнює їх із добовими цілями. Такий підхід дозволяє користувачу бачити не лише окремі записи, а й загальну картину за день. Водночас ці дані мають інформаційний характер і не є медичними або дієтологічними призначеннями.

Dashboard об’єднує результати роботи тренувального й харчового модулів. На цій сторінці користувач може побачити основні відомості про профіль, активний тренувальний план, виконані тренування, короткий підсумок харчування та рекомендації. Реалізація Dashboard дає змогу зібрати ключову інформацію в одному місці та не змушує користувача щоразу переходити між усіма сторінками системи.

Рекомендаційний модуль FitAI реалізовано на основі правил. Це означає, що система не навчає власну модель машинного навчання і не використовує нейронну мережу для прийняття основних рішень. Рекомендації формуються за наперед визначеними умовами, які аналізують фактичні дані користувача: профіль, активний тренувальний план, історію виконання тренувань, добові цілі харчування та записи журналу.

Прикладами умов для формування рекомендацій можуть бути: – відсутність заповненого профілю користувача; – відсутність активного тренувального плану; – наявність активного плану без виконаних тренувань; – наявність виконаних тренувань в історії; – відсутність добових цілей харчування; – відсутність записів у журналі харчування за поточний день; – відхилення фактичних харчових показників від установлених добових цілей.

Такий підхід робить рекомендаційний модуль пояснюваним. Кожна рекомендація може бути пов’язана з конкретною умовою, яка спрацювала під час аналізу даних. Наприклад, якщо користувач ще не створив тренувальний план, система може запропонувати сформувати його. Якщо журнал

харчування порожній, система може нагадати про необхідність внести дані за день. Якщо користувач регулярно позначає тренування як виконані, рекомендації можуть враховувати наявність позитивної активності.

OpenAI API використовується у FitAI як зовнішній сервіс для текстового узагальнення рекомендацій [24; 25; 26; 27]. Основна логіка рекомендацій спочатку виконується всередині системи на основі правил. Після цього backend може передати структурований набір рекомендацій до OpenAI API, щоб отримати короткий узагальнений текст у більш зручній для користувача формі.

Важливо, що OpenAI API не замінює рекомендаційний модуль FitAI і не приймає основних рішень щодо тренувань або харчування. Він використовується лише для мовного оформлення вже сформованих рекомендацій. Завдяки цьому система зберігає контрольованість, пояснюваність і відповідність фактичній реалізації.

Для захисту ключа OpenAI API звернення до зовнішнього сервісу виконується лише на серверній стороні. Ключ не передається до клієнтської частини React і не зберігається у відкритому frontend-коді. У разі недоступності зовнішнього сервісу основні функції FitAI мають залишатися працездатними: користувач усе одно може працювати з профілем, планом, історією, модулем харчування, Dashboard і базовими рекомендаціями на основі правил.

Таким чином, реалізація тренувального планування, модуля харчування та рекомендацій у FitAI побудована як взаємопов’язана система. Профіль користувача використовується для персоналізації плану, історія виконання тренувань відображає фактичну активність, модуль харчування фіксує введені користувачем показники, Dashboard узагальнює ключові дані, а рекомендаційний модуль формує пояснювані підказки на основі правил. Інтеграція з OpenAI API доповнює цю логіку текстовим AI-узагальненням, але не змінює основного принципу роботи системи.

3.5 Тестування основних функцій системи

Тестування вебплатформи FitAI виконувалося з метою перевірки працездатності основних функціональних сценаріїв, відповідності реалізованої системи сформульованим вимогам і виявлення можливих помилок у взаємодії між клієнтською частиною, серверною частиною та базою даних PostgreSQL [29; 30].

Оскільки FitAI є навчальним інженерним MVP, основний акцент зроблено на ручному функціональному тестуванні через інтерфейс користувача та перевірці типових сценаріїв роботи системи. Додатково окремі сценарії можуть перевірятися через API-запити до серверної частини, якщо потрібно підтвердити коректність роботи backend незалежно від клієнтського інтерфейсу.

Об’єктами тестування були такі функціональні модулі: – реєстрація та авторизація користувача; – захист приватних сторінок за допомогою JWT; – створення та редагування профілю користувача; – перегляд і фільтрація бази вправ; – генерація персонального тренувального плану; – перегляд тренувального плану по днях; – позначення тренування як виконаного; – перегляд історії виконаних тренувань; – створення та редагування добових цілей харчування; – додавання записів до журналу харчування; – формування підсумку харчування за день; – відображення Dashboard; – формування рекомендацій на основі правил; – генерація текстового AI-узагальнення через OpenAI API; – коректна робота базових рекомендацій у разі недоступності зовнішнього AI-сервісу.

Тестування проводилося у локальному або демонстраційному середовищі, у якому клієнтська частина React/Vite взаємодіє із серверною частиною Node.js / Express, а серверна частина підключається до бази даних PostgreSQL. Перед початком перевірки необхідно запустити backend, frontend і переконатися в наявності підключення до бази даних.

Основні тестові сценарії наведено в таблиці 3.1.

Таблиця 3.1 – Тестові сценарії перевірки основних функцій FitAI

Код	Сценарій тестування	Дії користувача / умови перевірки	Очікуваний результат
ТС-01	Реєстрація нового користувача	Відкрити сторінку реєстрації, ввести коректні дані та надіслати форму	Новий користувач створюється в системі, після чого користувач може перейти до авторизації
ТС-02	Перевірка помилки під час реєстрації	Надіслати форму реєстрації з порожніми або некоректними даними	Система не створює некоректний обліковий запис і показує повідомлення про помилку
ТС-03	Авторизація користувача	Ввести коректний email і пароль на сторінці входу	Сервер повертає JWT-токен, користувач отримує доступ до приватних сторінок
ТС-04	Авторизація з некоректними даними	Ввести неправильний пароль або неіснуючі облікові дані	Система не надає доступ і показує повідомлення про помилку авторизації
ТС-05	Захист приватних сторінок	Спробувати відкрити Dashboard, профіль або NutritionPage без авторизації	Доступ до приватної сторінки не надається, користувач має авторизуватися
ТС-06	Створення або редагування профілю	Після авторизації відкрити профіль, ввести або змінити параметри користувача та зберегти їх	Дані профілю зберігаються в PostgreSQL і повторно відображаються після оновлення сторінки
ТС-07	Перегляд бази вправ	Відкрити сторінку вправ після авторизації	Система відображає перелік вправ, отриманий із серверної частини
ТС-08	Фільтрація вправ	Застосувати доступні фільтри на сторінці вправ	Список вправ змінюється відповідно до вибраних параметрів
ТС-09	Генерація тренувального плану	За наявності заповненого профілю запустити генерацію персонального плану	Система створює тренувальний план і зберігає його в базі даних
ТС-10	Перегляд плану по днях	Відкрити сторінку активного тренувального плану	План відображається у структурованому вигляді по днях із переліком вправ

Код	Сценарій тестування	Дії користувача / умови перевірки	Очікуваний результат
ТС-11	Позначення тренування як виконаного	На сторінці плану позначити тренування як виконане	У базі даних створюється запис про виконане тренування, а інформація враховується в історії
ТС-12	Перегляд історії тренувань	Відкрити сторінку історії після виконання одного або кількох тренувань	Система відображає перелік виконаних тренувань поточного користувача
ТС-13	Створення добових цілей харчування	Відкрити NutritionPage, ввести цілі щодо калорій, білків, жирів, вуглеводів і води та зберегти їх	Добові цілі зберігаються та відображаються користувачу
ТС-14	Додавання запису до журналу харчування	Додати запис харчування із зазначенням калорій, білків, жирів, вуглеводів і води	Запис додається до журналу за обрану дату
ТС-15	Формування підсумку харчування за день	Додати один або кілька записів до журналу харчування	Система підраховує сумарні значення за день і порівнює їх із добовими цілями
ТС-16	Відображення Dashboard	Після авторизації відкрити Dashboard	Система відображає зведені дані про профіль, активний план, тренувальний прогрес, харчування та рекомендації
ТС-17	Формування рекомендацій на основі правил	Мати заповнений профіль, активний план, історію або харчові записи й відкрити блок рекомендацій	Система формує рекомендації відповідно до внутрішніх правил
ТС-18	Рекомендації за відсутності частини даних	Відкрити рекомендації без активного плану або без харчових записів	Система формує пояснювану рекомендацію щодо необхідності створити план або заповнити відповідні дані
ТС-19	AI-узагальнення рекомендацій	Отримати rule-based рекомендації та запустити текстове AI-узагальнення	Система звертається до backend, який використовує OpenAI API для формування текстового підсумку

Код	Сценарій тестування	Дії користувача / умови перевірки	Очікуваний результат
ТС-20	Робота без AI-узагальнення	Перевірити поведінку системи у разі помилки або недоступності OpenAI API	Основні функції FitAI та базові рекомендації на основі правил залишаються доступними

Під час тестування особлива увага приділялася перевірці зв’язку між клієнтською та серверною частинами. Для цього перевірялося, чи надсилає React-застосунок коректні HTTP-запити до REST API, чи додається JWT-токен до захищених запитів, чи правильно backend визначає поточного користувача та чи повертає лише його власні дані.

Окремо перевірялася робота з PostgreSQL. Після створення або зміни даних у клієнтському інтерфейсі виконувалася повторна перевірка їх відображення після оновлення сторінки або повторного входу. Така перевірка дає змогу підтвердити, що дані не зберігаються лише у стані клієнтської частини, а записуються до бази даних.

Під час перевірки модуля харчування оцінювалася коректність збереження добових цілей, додавання записів журналу та формування підсумку за день. Важливо, що цей модуль тестувався як інформаційний інструмент обліку введених користувачем даних, а не як медична або дієтологічна система.

Під час тестування рекомендаційного модуля перевірялося, чи формуються рекомендації на основі фактичних даних користувача. Основними умовами для перевірки були наявність або відсутність профілю, активного тренувального плану, історії виконання тренувань, добових цілей харчування та записів журналу. Такий підхід дає змогу підтвердити, що рекомендації мають пояснювану rule-based логіку.

Окремий сценарій стосувався AI-узагальнення. Перевірялося, що OpenAI API використовується лише як зовнішній сервіс для текстового узагальнення вже сформованих рекомендацій. Основна логіка рекомендацій не повинна залежати від відповіді зовнішнього сервісу. У разі недоступності

OpenAI API користувач усе одно має отримувати базові рекомендації, сформовані системою правил.

За результатами тестування можна зробити висновок, що набір перевірених сценаріїв охоплює основну функціональність вебплатформи FitAI: автентифікацію, роботу з профілем, вправами, тренувальним планом, історією, харчуванням, Dashboard, рекомендаціями та AI-узагальненням. Такий обсяг тестування є достатнім для підтвердження працездатності навчального MVP і демонстрації відповідності реалізованої системи основним функціональним вимогам.

3.6 Демонстрація роботи вебзастосунку

Демонстрація роботи вебплатформи FitAI виконується для підтвердження працездатності основних функціональних модулів системи та наочного показу взаємодії користувача з розробленим програмним продуктом. Демонстраційний сценарій охоплює ключові етапи роботи з вебзастосунком: реєстрацію, авторизацію, заповнення профілю, перегляд вправ, генерацію тренувального плану, фіксацію виконаного тренування, перегляд історії, роботу з модулем харчування, Dashboard, рекомендаціями та AI-узагальненням.

Перед початком демонстрації необхідно запустити серверну частину Node.js / Express, клієнтську частину React / Vite та переконатися в наявності підключення до бази даних PostgreSQL. Серверна частина приймає запити від клієнтської частини через REST API, працює з базою даних, перевіряє JWT-токен і формує відповіді у форматі JSON. Клієнтська частина відображає отримані дані в інтерфейсі користувача.

Першим етапом демонстрації є реєстрація нового користувача. Користувач відкриває сторінку реєстрації, вводить необхідні дані та надсилає форму. Після цього серверна частина створює новий обліковий запис у базі даних. На цьому етапі демонструється можливість створення користувача та взаємодія клієнтської частини з backend.

Другим етапом є авторизація. Користувач вводить облікові дані на сторінці входу. Якщо дані коректні, сервер повертає JWT-токен, який надалі використовується для доступу до приватних сторінок вебплатформи. Після входу користувач отримує доступ до Dashboard, профілю, тренувального плану, історії, NutritionPage і рекомендацій.

Третім етапом демонстрації є робота з профілем користувача. Користувач відкриває сторінку профілю, вводить або змінює персональні параметри, зокрема рівень фізичної підготовки та тренувальну ціль, після чого зберігає зміни. Ці дані записуються до PostgreSQL і надалі використовуються для генерації тренувального плану та формування рекомендацій.

Четвертим етапом є перегляд бази вправ. Користувач відкриває сторінку вправ і переглядає перелік доступних вправ, отриманих із серверної частини. За потреби застосовується фільтрація за передбаченими параметрами. Цей етап демонструє роботу модуля вправ і отримання структурованих даних із бази даних.

П’ятим етапом є генерація персонального тренувального плану. Система використовує дані профілю користувача та базу вправ для формування плану. Після генерації план зберігається у PostgreSQL і відображається користувачу по днях. У межах демонстрації показується, що план не зникає після оновлення сторінки, а зберігається як частина персональних даних користувача.

Шостим етапом є фіксація виконаного тренування. Користувач відкриває тренувальний план, обирає тренування та позначає його як виконане. Після цього серверна частина створює запис в історії тренувань. На сторінці історії користувач може переглянути виконані тренування. Цей етап демонструє зв’язок між планом, дією користувача та історією виконання.

Сьомим етапом є робота з модулем харчування. Користувач відкриває NutritionPage, створює або редагує добові цілі харчування, зокрема калорії, білки, жири, вуглеводи та воду. Після цього користувач додає записи до журналу харчування за конкретну дату. Система формує підсумок за день і

порівнює фактичні значення з установленими цілями. Цей модуль має інформаційний характер і не подається як медична або дієтологічна система.

Восьмим етапом демонстрації є перегляд Dashboard. На цій сторінці користувач бачить узагальнену інформацію про свій профіль, активний тренувальний план, тренувальний прогрес, харчові показники та рекомендації. Dashboard виконує роль центральної сторінки, яка об'єднує дані з кількох модулів системи.

Дев'ятим етапом є демонстрація рекомендаційного модуля. Система аналізує профіль користувача, активний тренувальний план, історію виконаних тренувань, добові цілі харчування та записи журналу. На основі цих даних формуються рекомендації за внутрішніми правилами. Наприклад, якщо користувач не має активного плану, система може запропонувати створити його; якщо журнал харчування порожній, система може нагадати про необхідність додати записи за день.

Десятим етапом є демонстрація AI-узагальнення. Після формування базових рекомендацій backend може передати структурований контекст до OpenAI API для створення короткого текстового підсумку. Важливо, що OpenAI API використовується лише як зовнішній сервіс для мовного узагальнення вже сформованих рекомендацій. Основна логіка рекомендацій залишається rule-based і виконується всередині FitAI.

Під час демонстрації також доцільно показати обробку типових ситуацій: спробу доступу до приватної сторінки без авторизації, повідомлення про відсутність активного тренувального плану, порожній журнал харчування або недоступність AI-узагальнення. Такі приклади підтверджують, що система не лише виконує основні сценарії, а й коректно реагує на неповні або помилкові умови роботи.

Отже, демонстрація FitAI підтверджує, що розроблена вебплатформа забезпечує повний базовий сценарій роботи користувача: від створення облікового запису до отримання персоналізованого тренувального плану, ведення історії тренувань, фіксації харчових показників, перегляду Dashboard,

отримання рекомендацій і текстового AI-узагальнення. Це свідчить про функціональну завершеність навчального MVP і відповідність реалізованого продукту поставленим завданням.

Висновки до розділу 3

У третьому розділі було розглянуто практичну реалізацію та тестування вебплатформи FitAI. Описано реалізацію серверної частини на основі Node.js / Express, клієнтської частини на основі React / Vite, роботу з базою даних PostgreSQL, реалізацію тренувального планування, модуля харчування, Dashboard, рекомендаційного модуля та текстового AI-узагальнення через зовнішній сервіс OpenAI API.

Серверна частина забезпечує обробку запитів клієнтської частини, авторизацію користувачів за допомогою JWT, роботу із захищеними маршрутами, взаємодію з PostgreSQL та формування відповідей у форматі JSON. Реалізовані backend-модулі відповідають за реєстрацію, вхід користувача, профіль, вправи, тренувальні плани, історію виконаних тренувань, харчування, Dashboard, рекомендації та AI-узагальнення.

Клієнтська частина реалізує інтерфейс користувача та основні сценарії взаємодії з вебплатформою. Користувач може зареєструватися, авторизуватися, заповнити профіль, переглядати базу вправ, формувати та переглядати тренувальний план, позначати тренування як виконані, переглядати історію, вести журнал харчування, переглядати Dashboard і отримувати рекомендації.

PostgreSQL використано як основне сховище даних вебплатформи. У базі даних зберігаються облікові записи користувачів, профілі, вправи, тренувальні плани, історія виконаних тренувань, добові цілі харчування, записи журналу харчування та дані, необхідні для формування рекомендацій. Взаємодія з базою даних здійснюється лише через серверну частину, що дає змогу контролювати доступ до персональних даних користувача.

Реалізований модуль тренувального планування забезпечує формування персонального плану на основі профілю користувача та бази вправ. План

зберігається в базі даних і відображається по днях. Модуль історії тренувань дає змогу фіксувати виконані тренування та використовувати ці дані для відображення прогресу й формування рекомендацій.

Модуль харчування забезпечує створення та редагування добових цілей, додавання записів до журналу харчування, фіксацію калорій, білків, жирів, вуглеводів і води, а також формування підсумку за день. При цьому модуль має інформаційний характер і не розглядається як медична або дієтологічна система.

Рекомендаційний модуль реалізовано як пояснювану систему на основі правил. Він аналізує профіль користувача, активний тренувальний план, історію виконання тренувань, добові цілі харчування та записи журналу. OpenAI API використовується як зовнішній сервіс для текстового узагальнення вже сформованих рекомендацій, а не як власна нейронна мережа FitAI чи основний механізм прийняття рішень.

У межах тестування перевірено основні функціональні сценарії роботи системи: реєстрацію, авторизацію, захист приватних сторінок, роботу з профілем, вправами, тренувальним планом, історією, модулем харчування, Dashboard, рекомендаціями та AI-узагальненням. Окремо враховано сценарій, за якого зовнішній AI-сервіс є недоступним: базові рекомендації на основі правил мають залишатися працездатними.

Демонстрація роботи вебплатформи підтверджує, що FitAI забезпечує повний базовий сценарій використання: від створення облікового запису до формування персонального тренувального плану, фіксації тренувального прогресу, ведення журналу харчування, перегляду Dashboard, отримання рекомендацій і текстового AI-узагальнення. Отже, реалізований програмний продукт відповідає поставленим завданням навчального інженерного MVP та може бути використаний як основа для подальшого розвитку вебплатформи.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено вебплатформу FitAI, призначену для формування персональних тренувальних планів, ведення історії тренувань, фіксації харчових показників і надання рекомендацій користувачу. Розроблений програмний продукт поєднує клієнтську частину React / Vite, серверну частину Node.js / Express, базу даних PostgreSQL, JWT-авторизацію, рекомендаційний модуль на основі правил та допоміжне AI-узагальнення через зовнішній сервіс OpenAI API.

У першому розділі було розглянуто предметну область цифрових фітнес-рішень, проаналізовано потреби користувачів і наявні програмні аналоги. Установлено, що значна частина наявних рішень або зосереджена переважно на тренуваннях, або на харчуванні, або на загальному обліку фізичної активності. Це підтвердило доцільність створення вебплатформи, яка поєднує тренувальне планування, журнал харчування, історію виконання тренувань і рекомендації в межах одного програмного середовища.

У другому розділі було сформульовано функціональні та нефункціональні вимоги до FitAI, спроєктовано загальну архітектуру системи, структуру бази даних PostgreSQL, REST API та логіку рекомендаційного модуля. Визначено основні модулі системи: автентифікація, профіль користувача, база вправ, тренувальні плани, історія виконаних тренувань, модуль харчування, Dashboard, рекомендації та AI-узагальнення. Окрему увагу приділено коректному позиціонуванню AI-функціональності: FitAI не є власною нейронною мережею або самонавчальною ML-моделлю, а використовує OpenAI API як зовнішній сервіс для текстового узагальнення вже сформованих рекомендацій.

У третьому розділі було описано практичну реалізацію вебплатформи. Серверна частина на основі Node.js / Express забезпечує обробку запитів, авторизацію користувачів, взаємодію з PostgreSQL, формування тренувальних планів, роботу з харчовим модулем, Dashboard, рекомендаціями та AI-

узагальненням. Клієнтська частина на основі React / Vite забезпечує користувацький інтерфейс і взаємодію з backend через REST API.

У межах реалізації було забезпечено реєстрацію та авторизацію користувача з використанням JWT, створення й редагування профілю, перегляд і фільтрацію бази вправ, генерацію персонального тренувального плану, перегляд плану по днях, позначення тренувань як виконаних і перегляд історії тренувань. Також реалізовано модуль харчування, який дає змогу створювати або редагувати добові цілі, додавати записи до журналу харчування, фіксувати калорії, білки, жири, вуглеводи та воду, а також переглядати підсумок за день.

Рекомендаційний модуль FitAI реалізовано як пояснювану систему на основі правил. Він аналізує профіль користувача, активний тренувальний план, історію виконаних тренувань, добові цілі харчування та записи журналу харчування. Такий підхід дає змогу формувати зрозумілі рекомендації, пов’язані з конкретними даними користувача. OpenAI API використовується лише як допоміжний інструмент для текстового узагальнення рекомендацій і не замінює основну логіку системи.

Під час тестування було перевірено основні сценарії роботи FitAI: реєстрацію, авторизацію, захист приватних сторінок, роботу з профілем, базою вправ, тренувальним планом, історією тренувань, модулем харчування, Dashboard, рекомендаціями та AI-узагальненням. Окремо враховано ситуацію недоступності зовнішнього AI-сервісу: базові рекомендації на основі правил мають залишатися доступними користувачу. Це підтверджує, що OpenAI API є допоміжним, а не критичним елементом працездатності системи.

Розроблений модуль харчування має інформаційний характер. Він призначений для фіксації введених користувачем показників, формування добових підсумків і порівняння фактичних значень із установленими цілями. Система не є медичною або дієтологічною платформою, а її рекомендації не повинні розглядатися як призначення спеціаліста.

Практичне значення роботи полягає в тому, що створена вебплатформа може бути використана як навчальний інженерний MVP для демонстрації повного циклу розробки сучасного вебзастосунку: від аналізу предметної області й проєктування архітектури до реалізації, тестування та демонстрації роботи. FitAI поєднує кілька взаємопов’язаних модулів і демонструє використання актуального технологічного стеку для створення персоналізованого вебсервісу.

Поставлену мету кваліфікаційної роботи досягнуто. Було спроєктовано й реалізовано вебплатформу FitAI, яка забезпечує базовий сценарій роботи користувача: створення облікового запису, заповнення профілю, формування персонального тренувального плану, фіксацію тренувального прогресу, ведення журналу харчування, перегляд Dashboard, отримання рекомендацій і текстового AI-узагальнення. Подальший розвиток системи може передбачати розширення бази вправ, покращення адаптивності інтерфейсу, додавання детальнішої аналітики прогресу, розширення можливостей модуля харчування та підготовку системи до повноцінного серверного розгортання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React Documentation. URL: https://react.dev/ (дата звернення: 03.06.2026).

2. Vite Documentation. URL: https://vitejs.dev/guide/ (дата звернення: 03.06.2026).

3. React Router Documentation. URL: https://reactrouter.com/ (дата звернення: 03.06.2026).

4. Axios Documentation. URL: https://axios-http.com/docs/intro (дата звернення: 03.06.2026).

5. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. Sebastopol : O'Reilly Media, 2020. 310 p.

6. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 03.06.2026).

7. Express Documentation. URL: https://expressjs.com/ (дата звернення: 03.06.2026).

8. Mardan A. Practical Node.js: Building Real-World Scalable Web Apps. 2nd ed. Berkeley : Apress, 2018. 398 p.

9. PostgreSQL 16 Documentation. The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/ (дата звернення: 03.06.2026).

10. Date C. J. An Introduction to Database Systems. 8th ed. Boston : Pearson, 2003. 1024 p.

11. Коннолл Т., Бегг К. Бази даних. Проєктування, реалізація та супровід. Київ : Діалектика, 2017. 1440 с.

12. node-postgres (pg) Documentation. URL: https://node-postgres.com/ (дата звернення: 03.06.2026).

13. SQL Tutorial. URL: https://www.postgresqltutorial.com/ (дата звернення: 03.06.2026).

14. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force, 2015. DOI: 10.17487/RFC7519.

15. jsonwebtoken. GitHub repository. URL: https://github.com/auth0/node-jsonwebtoken (дата звернення: 03.06.2026).

16. bcrypt for Node.js. GitHub repository. URL: https://github.com/kelektiv/node.bcrypt.js (дата звернення: 03.06.2026).

17. express-validator Documentation. URL: https://express-validator.github.io/docs/ (дата звернення: 03.06.2026).

18. CORS middleware for Express. URL: https://github.com/expressjs/cors (дата звернення: 03.06.2026).

19. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Ph.D. dissertation. University of California, Irvine, 2000. 162 p.

20. Richardson L., Ruby S. RESTful Web Services. Sebastopol : O'Reilly Media, 2007. 446 p.

21. MDN Web Docs. HTTP. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP (дата звернення: 03.06.2026).

22. Fielding R., Reschke J. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231. IETF, 2014.

23. OWASP Top 10:2021. The Ten Most Critical Web Application Security Risks. OWASP Foundation. URL: https://owasp.org/Top10/ (дата звернення: 03.06.2026).

24. OpenAI API Documentation. URL: https://platform.openai.com/docs (дата звернення: 03.06.2026).

25. OpenAI Node.js Library. GitHub repository. URL: https://github.com/openai/openai-node (дата звернення: 03.06.2026).

26. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Hoboken : Pearson, 2021. 1136 p.

27. Giarratano J., Riley G. Expert Systems: Principles and Programming. 4th ed. Boston : Course Technology, 2004. 856 p.

28. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol : O'Reilly Media, 2020. 706 p.

29. Соммервілл І. Інженерія програмного забезпечення. 10-те вид. Київ : Вільямс, 2020. 816 с.

30. ISO/IEC 25010:2023. Systems and software engineering – SQuaRE – Product quality model. Geneva : ISO, 2023.

ДОДАТКИ

Додаток А – Структура проєкту FitAI

У цьому додатку наведено загальну структуру програмного проєкту FitAI та посилання на репозиторій із вихідним кодом.

Посилання на репозиторій: https://github.com/tymsnit/fitAI .

Проєкт FitAI складається з клієнтської та серверної частин. Клієнтська частина реалізована з використанням React та Vite, а серверна частина – з використанням Node.js / Express. Для збереження даних використовується PostgreSQL. Backend працює зі змінними середовища, у яких зберігаються параметри підключення до бази даних, JWT secret, порт запуску серверної частини та ключ OpenAI API.

Таблиця А.1 – Основні складові проєкту FitAI

Елемент структури	Призначення
frontend / client	Клієнтська частина вебплатформи React / Vite
src	Основний каталог вихідного коду клієнтської частини
components	Повторно використовувані компоненти інтерфейсу
pages	Сторінки вебзастосунку: реєстрація, авторизація, Dashboard, профіль, вправи, план, історія, харчування, рекомендації
services / api	Модулі взаємодії клієнтської частини з REST API
backend / server	Серверна частина Node.js / Express
src	Основний каталог вихідного коду серверної частини
routes	Маршрути REST API
controllers	Обробники запитів
middleware	Проміжна логіка, зокрема перевірка JWT
services	Допоміжна бізнес-логіка, зокрема формування планів, рекомендацій та AI-узагальнення
db / config	Налаштування підключення до PostgreSQL
.env	Файл змінних середовища, який не повинен публікуватися в репозиторії
README.md	Опис проєкту, стеку технологій і порядку запуску
package.json	Налаштування залежностей і команд запуску

Наведена структура відображає поділ FitAI на клієнтську частину, серверну частину та базу даних. Такий поділ відповідає клієнт-серверній архітектурі вебплатформи та полегшує супровід програмного коду.

Додаток Б – Структура бази даних PostgreSQL

У цьому додатку наведено структуру бази даних PostgreSQL, яка використовується для збереження даних вебплатформи FitAI.

База даних містить таблиці для збереження облікових записів користувачів, профілів, вправ, тренувальних планів, історії виконаних тренувань, добових цілей харчування, записів журналу харчування та даних, необхідних для формування рекомендацій.

Таблиця Б.1 – Основні таблиці бази даних FitAI

Таблиця	Призначення
users	Збереження облікових записів користувачів
user_profiles	Збереження персональних параметрів користувача
exercises	Збереження бази вправ
workout_plans	Збереження персональних тренувальних планів
workouts	Збереження тренувальних днів або окремих тренувань у межах плану
workout_exercises	Зв’язок тренувань із вправами
workout_logs	Збереження історії виконаних тренувань
nutrition_targets	Збереження добових цілей харчування
nutrition_logs	Збереження записів журналу харчування
recommendations	Збереження або відображення сформованих рекомендацій

Основні зв’язки між таблицями бази даних:

Зв’язок	Опис
users → user_profiles	Один користувач має один профіль
users → workout_plans	Один користувач може мати один або кілька тренувальних планів
workout_plans → workouts	Один тренувальний план містить кілька тренувальних днів
workouts → workout_exercises	Одне тренування містить кілька вправ
exercises → workout_exercises	Одна вправа може використовуватися в різних тренуваннях
users → workout_logs	Один користувач може мати багато записів історії тренувань
users → nutrition_targets	Один користувач має добові цілі харчування

Зв’язок	Опис
users → nutrition_logs	Один користувач може мати багато записів журналу харчування
users → recommendations	Один користувач може мати набір рекомендацій

Додаток В – Основні API endpoint-и вебплатформи fitai

У цьому додатку наведено перелік основних endpoint-ів REST API, які забезпечують взаємодію клієнтської частини React із серверною частиною Node.js / Express.

Таблиця В.1 – Основні групи API endpoint-ів FitAI

Група endpoint-ів	Приклад endpoint-у	Метод	Призначення
Auth	/api/auth/register	POST	Реєстрація нового користувача
Auth	/api/auth/login	POST	Авторизація користувача та отримання JWT
Profile	/api/profile	GET	Отримання профілю поточного користувача
Profile	/api/profile	POST / PUT	Створення або оновлення профілю
Exercises	/api/exercises	GET	Отримання переліку вправ
Exercises	/api/exercises?filter=…	GET	Фільтрація вправ за параметрами
Workout plans	/api/workout-plans	POST	Генерація або створення тренувального плану
Workout plans	/api/workout-plans	GET	Отримання активного тренувального плану

Група endpoint-ів	Приклад endpoint-у	Метод	Призначення
Workout logs	/api/workout-logs	POST	Позначення тренування як виконаного
Workout logs	/api/workout-logs	GET	Отримання історії виконаних тренувань
Nutrition	/api/nutrition/targets	GET	Отримання добових цілей харчування
Nutrition	/api/nutrition/targets	POST / PUT	Створення або оновлення добових цілей харчування
Nutrition	/api/nutrition/logs	POST	Додавання запису до журналу харчування
Nutrition	/api/nutrition/logs	GET	Отримання записів журналу харчування за дату
Nutrition	/api/nutrition/summary	GET	Отримання підсумку харчування за день
Dashboard	/api/dashboard	GET	Отримання зведених даних для Dashboard
Recommendations	/api/recommendations	GET	Отримання рекомендацій на основі правил
AI summary	/api/recommendations/summary	POST	Отримання текстового AI-узагальнення рекомендацій

Усі endpoint-и, які працюють із персональними даними користувача, мають бути захищені JWT-авторизацією. Клієнтська частина передає JWT-токен у заголовку запиту, після чого backend визначає поточного користувача та повертає лише ті дані, які належать йому.

Додаток Г – Тестові сценарії та демонстраційні матеріали FitAI

У цьому додатку наведено тестові сценарії та перелік демонстраційних матеріалів, які підтверджують працездатність основних модулів вебплатформи FitAI.

Таблиця Г.1 – Тестові сценарії вебплатформи FitAI

Код	Сценарій	Очікуваний результат
ТС-01	Реєстрація нового користувача	Користувач створюється в системі
ТС-02	Авторизація користувача	Користувач отримує JWT і доступ до приватних сторінок
ТС-03	Спроба доступу без авторизації	Доступ до приватних сторінок блокується
ТС-04	Створення або редагування профілю	Дані профілю зберігаються та повторно відображаються
ТС-05	Перегляд бази вправ	Система відображає перелік вправ
ТС-06	Фільтрація вправ	Список вправ змінюється відповідно до вибраних параметрів
ТС-07	Генерація тренувального плану	Система створює персональний тренувальний план
ТС-08	Перегляд плану по днях	План відображається у структурованому вигляді
ТС-09	Позначення тренування як виконаного	У системі створюється запис історії тренування
ТС-10	Перегляд історії тренувань	Користувач бачить виконані тренування
ТС-11	Створення добових цілей харчування	Цілі харчування зберігаються в системі
ТС-12	Додавання запису до журналу харчування	Запис додається до журналу за обрану дату
ТС-13	Формування підсумку харчування за день	Система підраховує фактичні значення за день
ТС-14	Відображення Dashboard	Користувач бачить зведені дані за основними модулями

Код	Сценарій	Очікуваний результат
ТС-15	Формування рекомендацій	Система формує рекомендації на основі правил
ТС-16	AI-узагальнення рекомендацій	Система формує текстовий підсумок через backend і OpenAI API
ТС-17	Недоступність OpenAI API	Базові рекомендації на основі правил залишаються доступними

Демонстраційні матеріали.

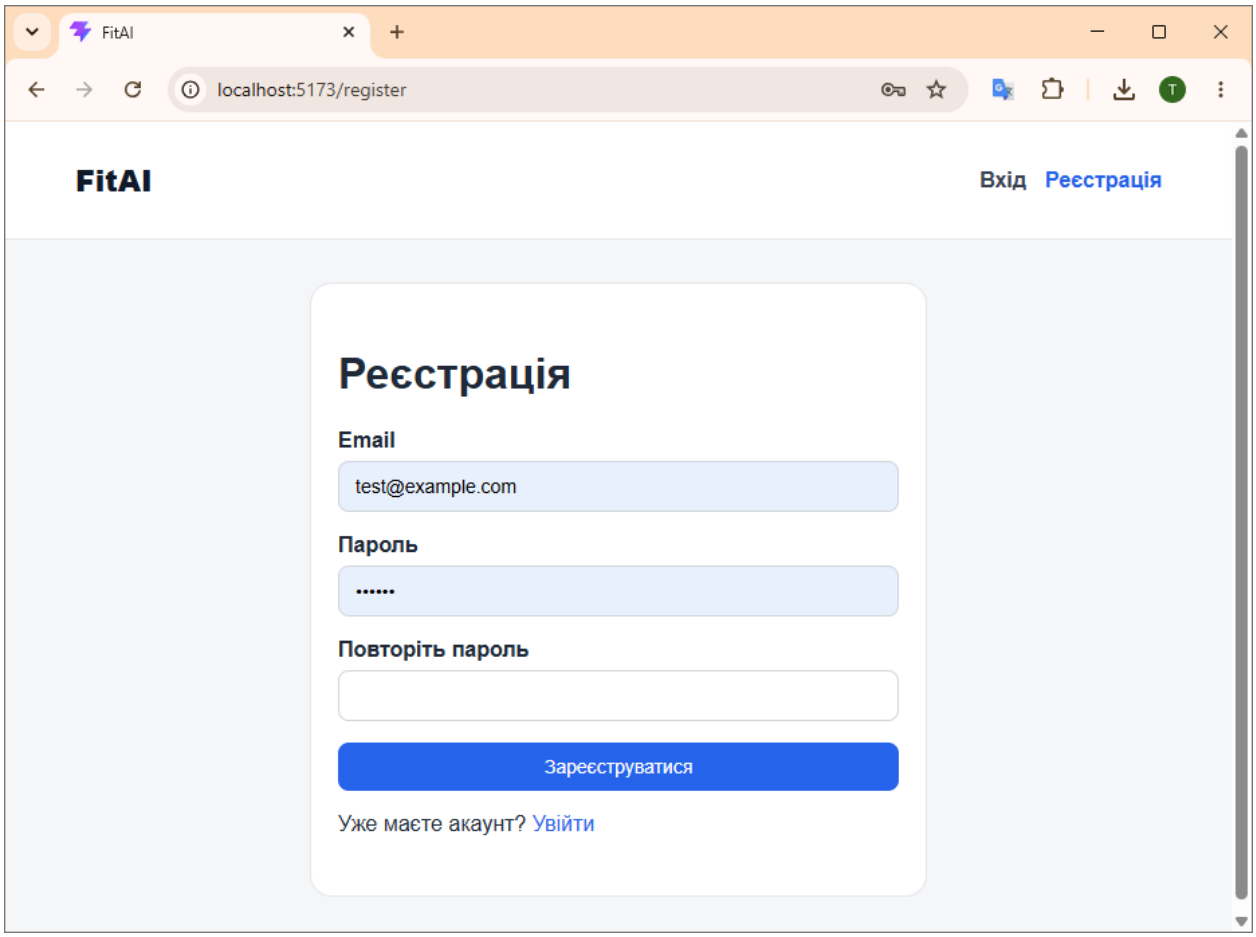

Рисунок Г.1 – Сторінка реєстрації користувача FitAI.

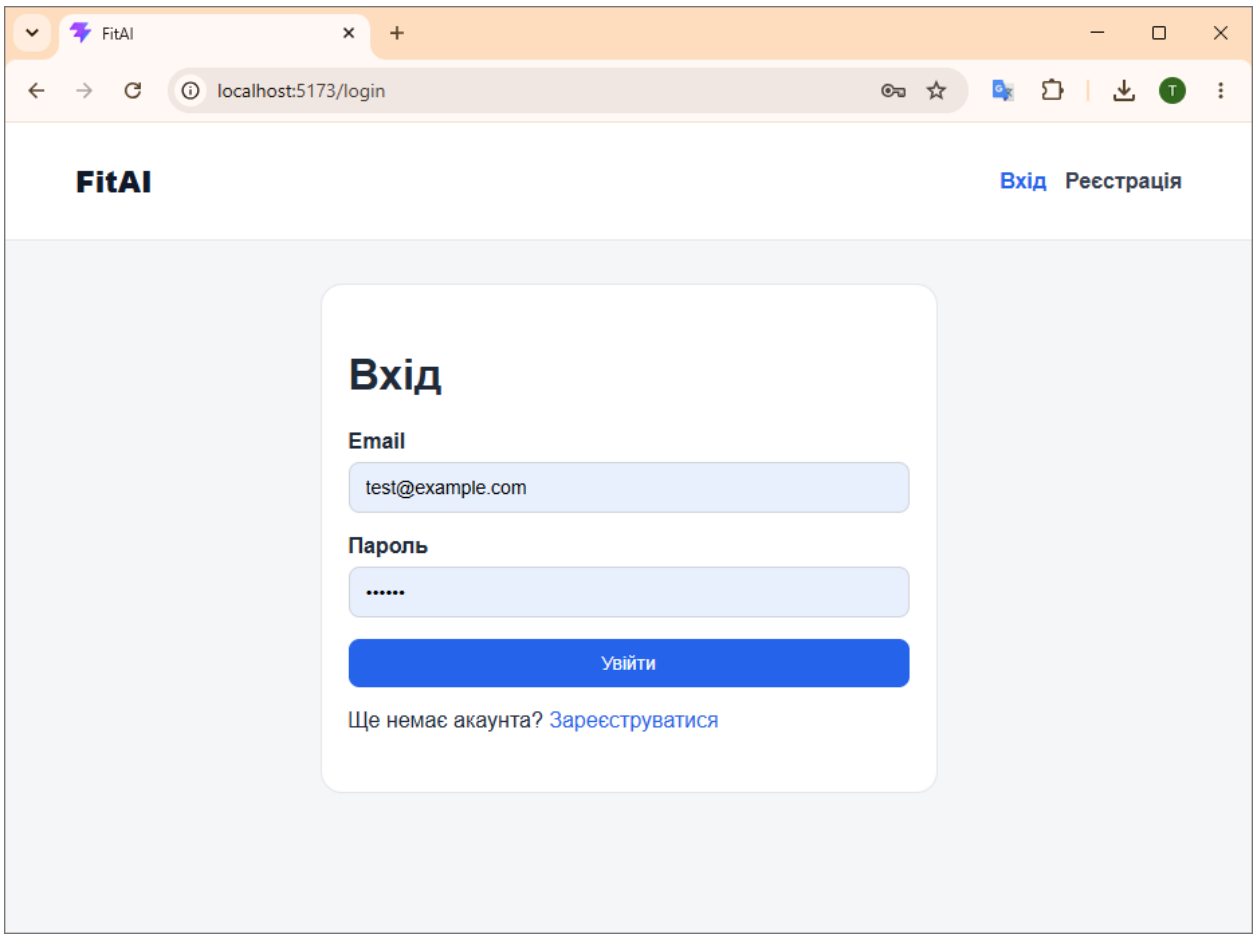

Рисунок Г.2 – Сторінка авторизації користувача FitAI.

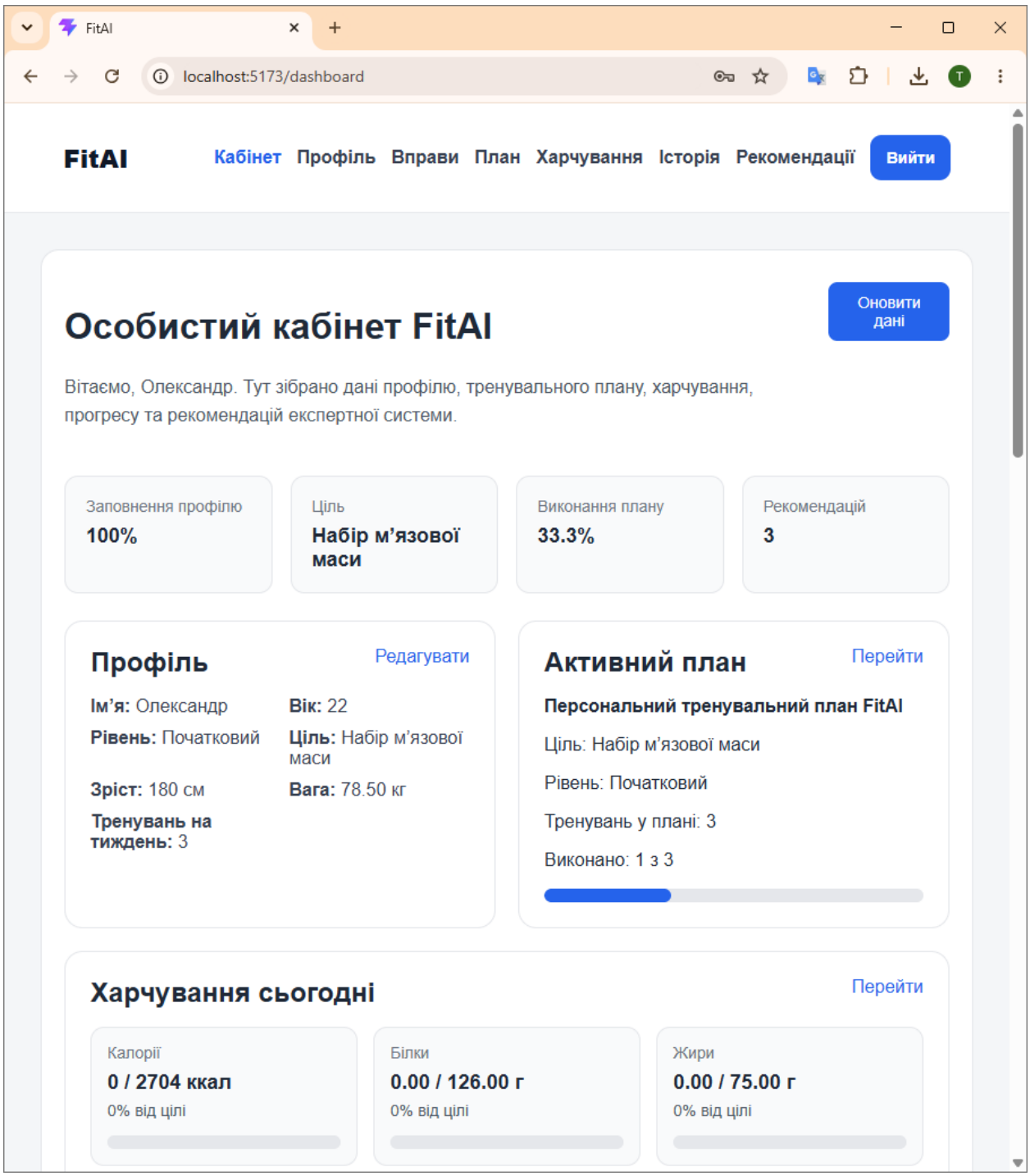

Рисунок Г.3 – Dashboard користувача.

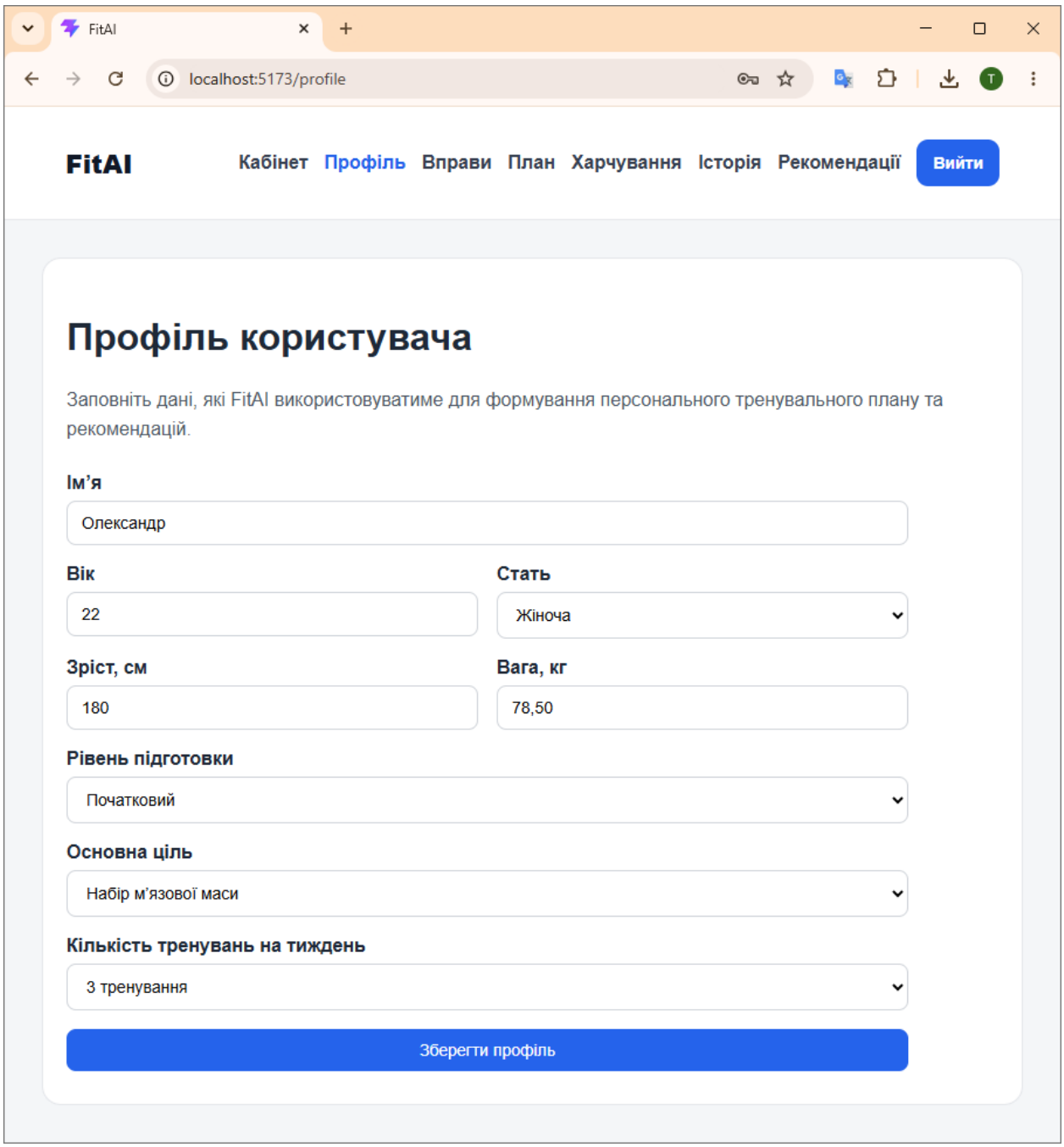

Рисунок Г.4 – Сторінка профілю користувача.

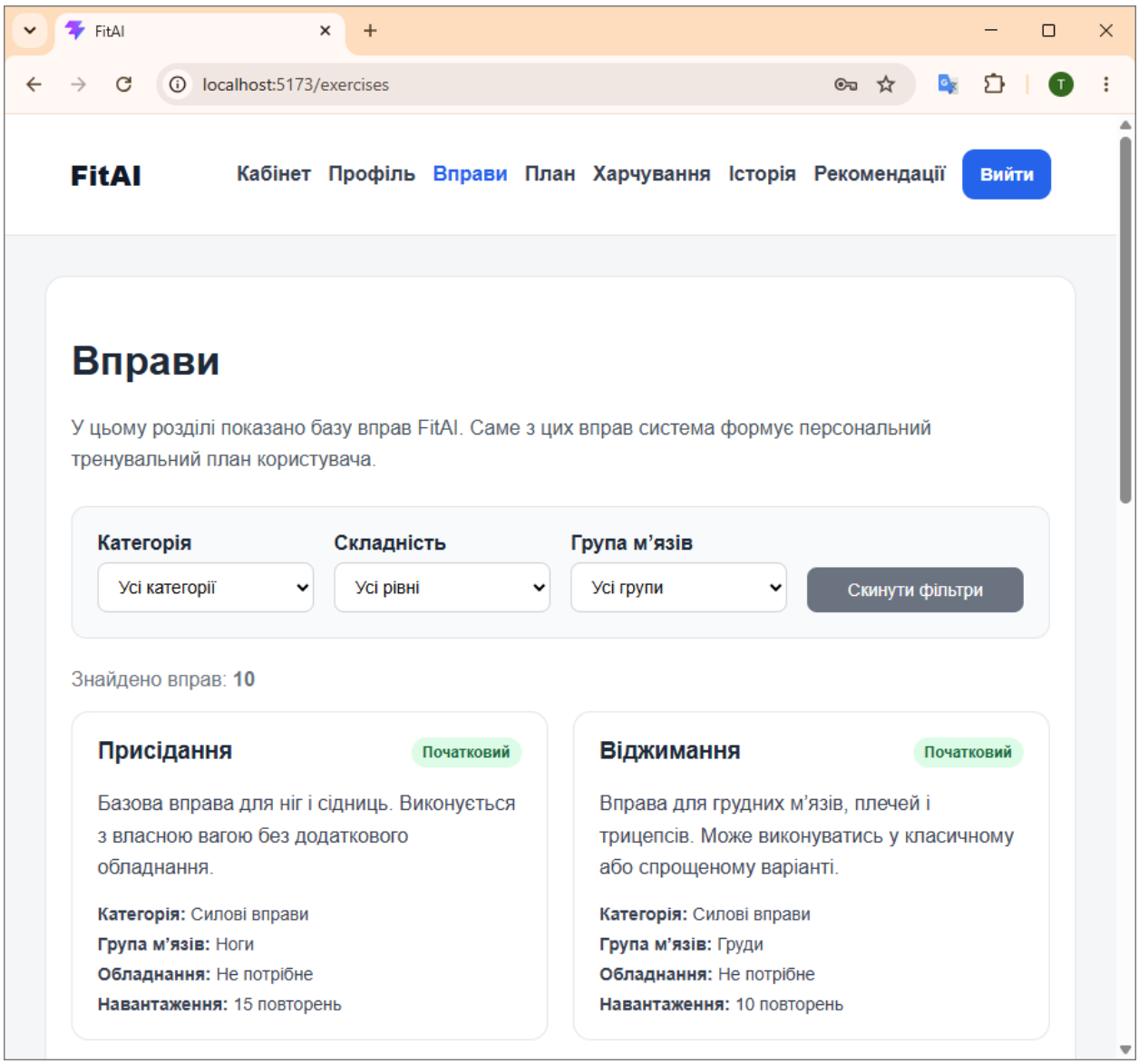

Рисунок Г.5 – База вправ та фільтрація вправ.

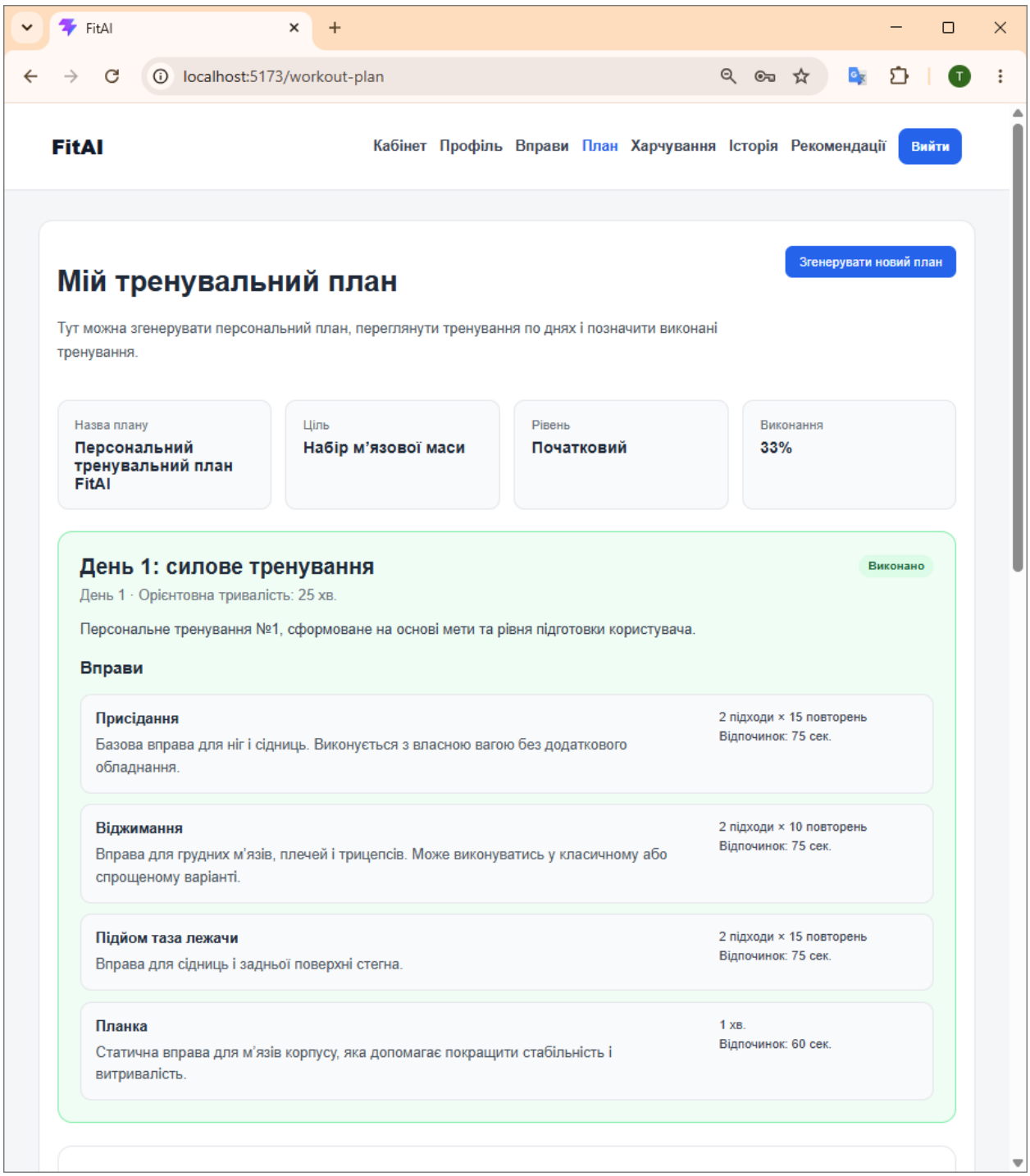

Рисунок Г.6 – Персональний тренувальний план по днях.

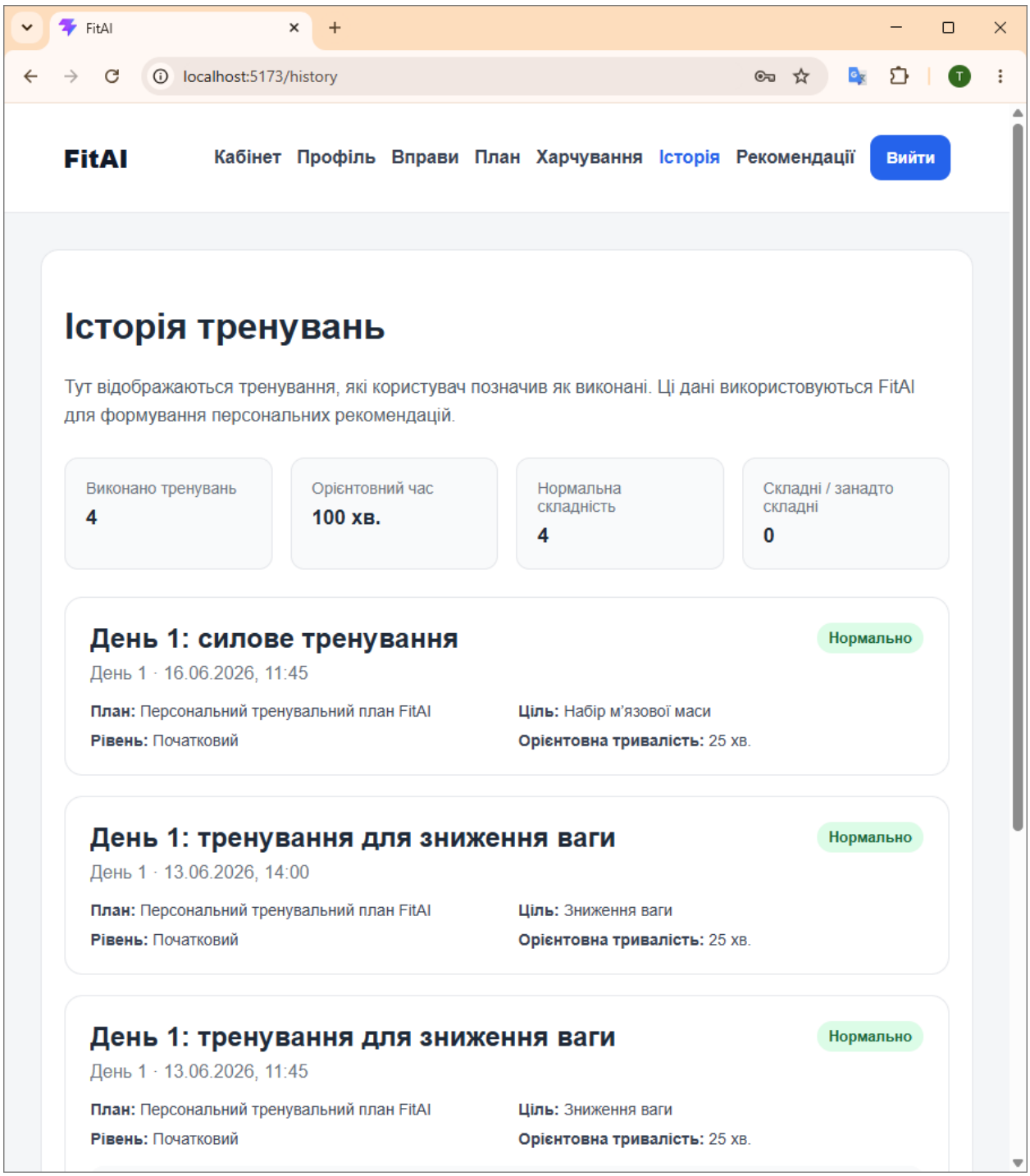

Рисунок Г.7 – Історія виконаних тренувань.

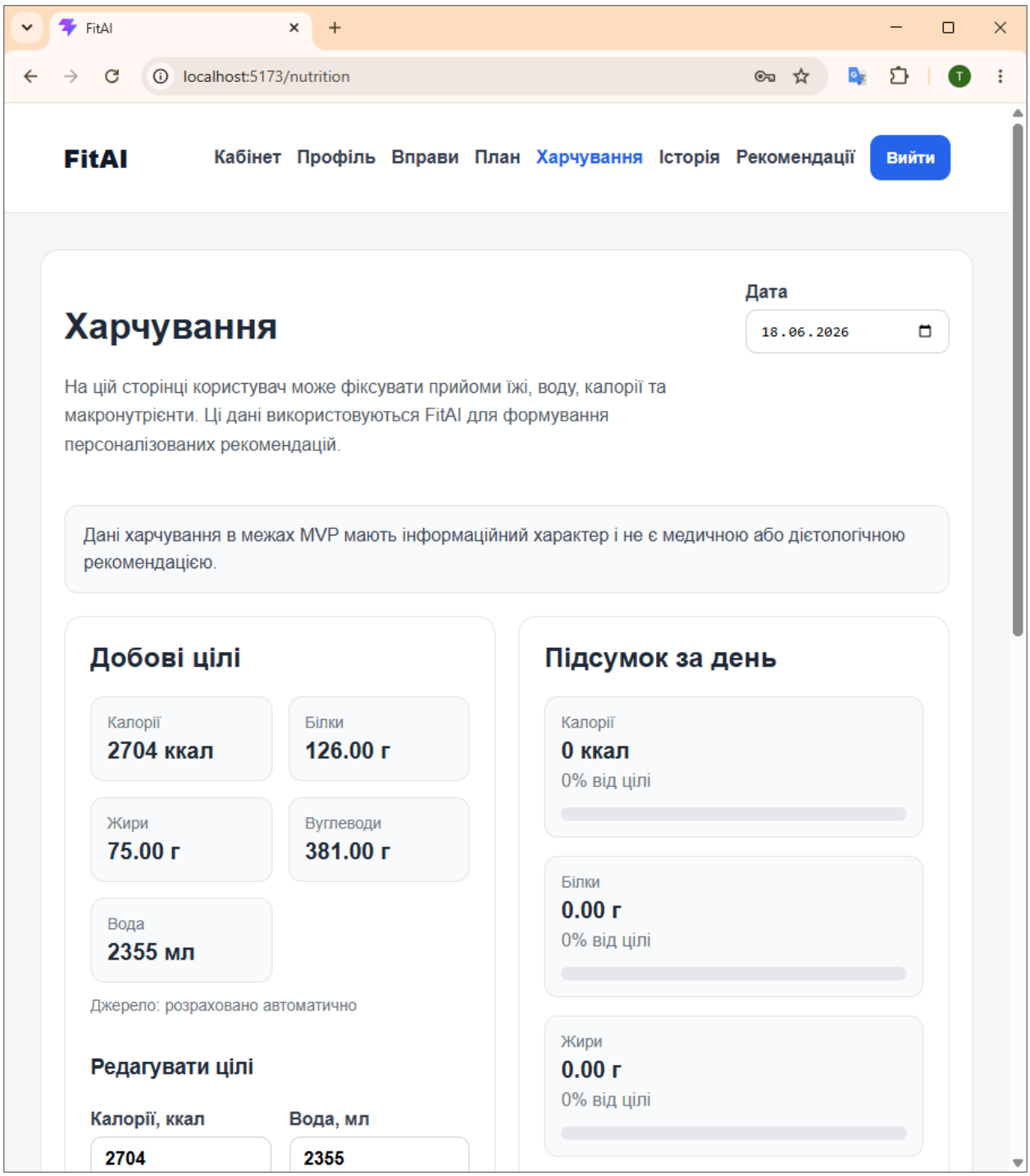

Рисунок Г.8 – NutritionPage: добові цілі та журнал харчування.

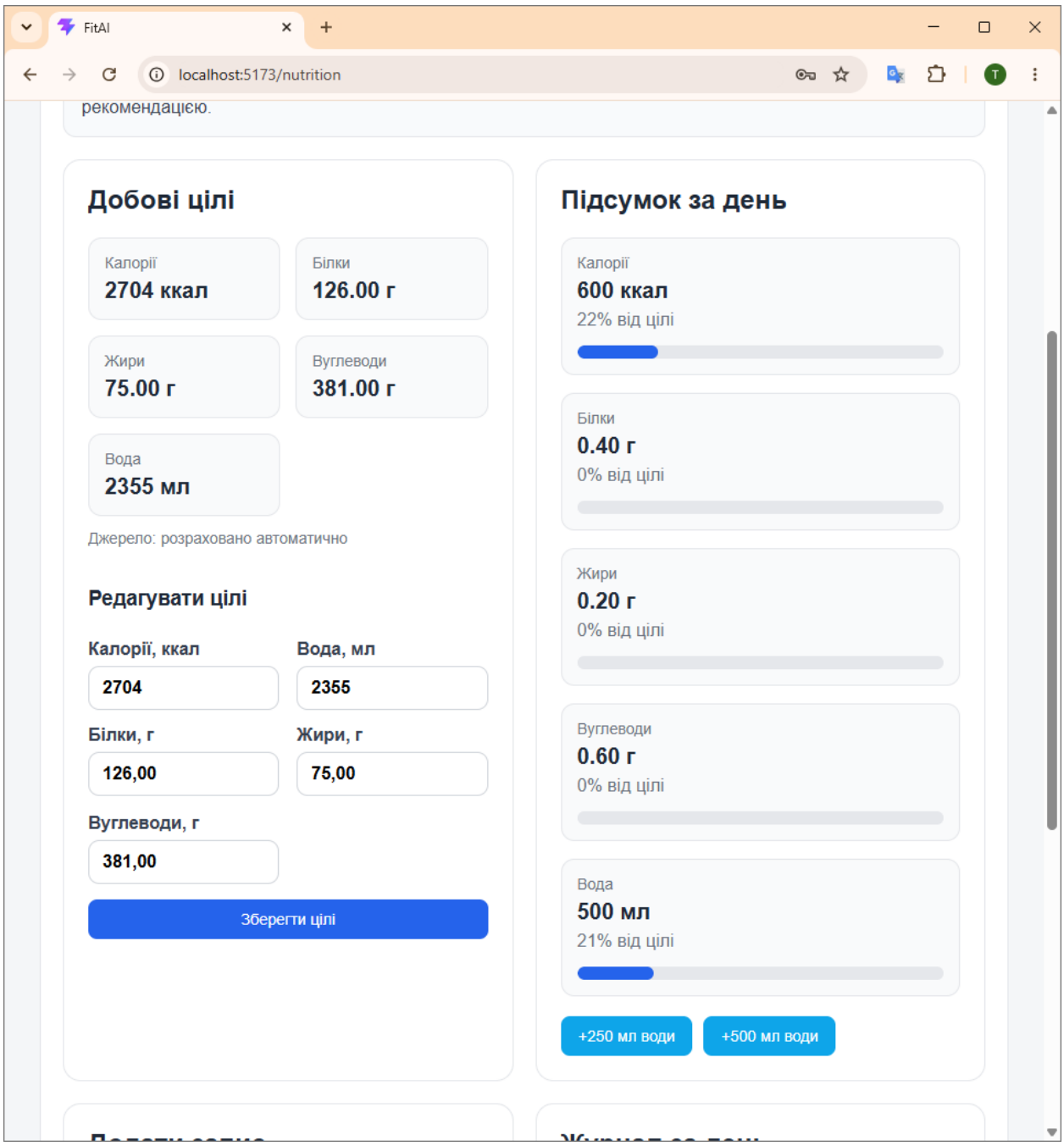

Рисунок Г.9 – Підсумок харчування за день.

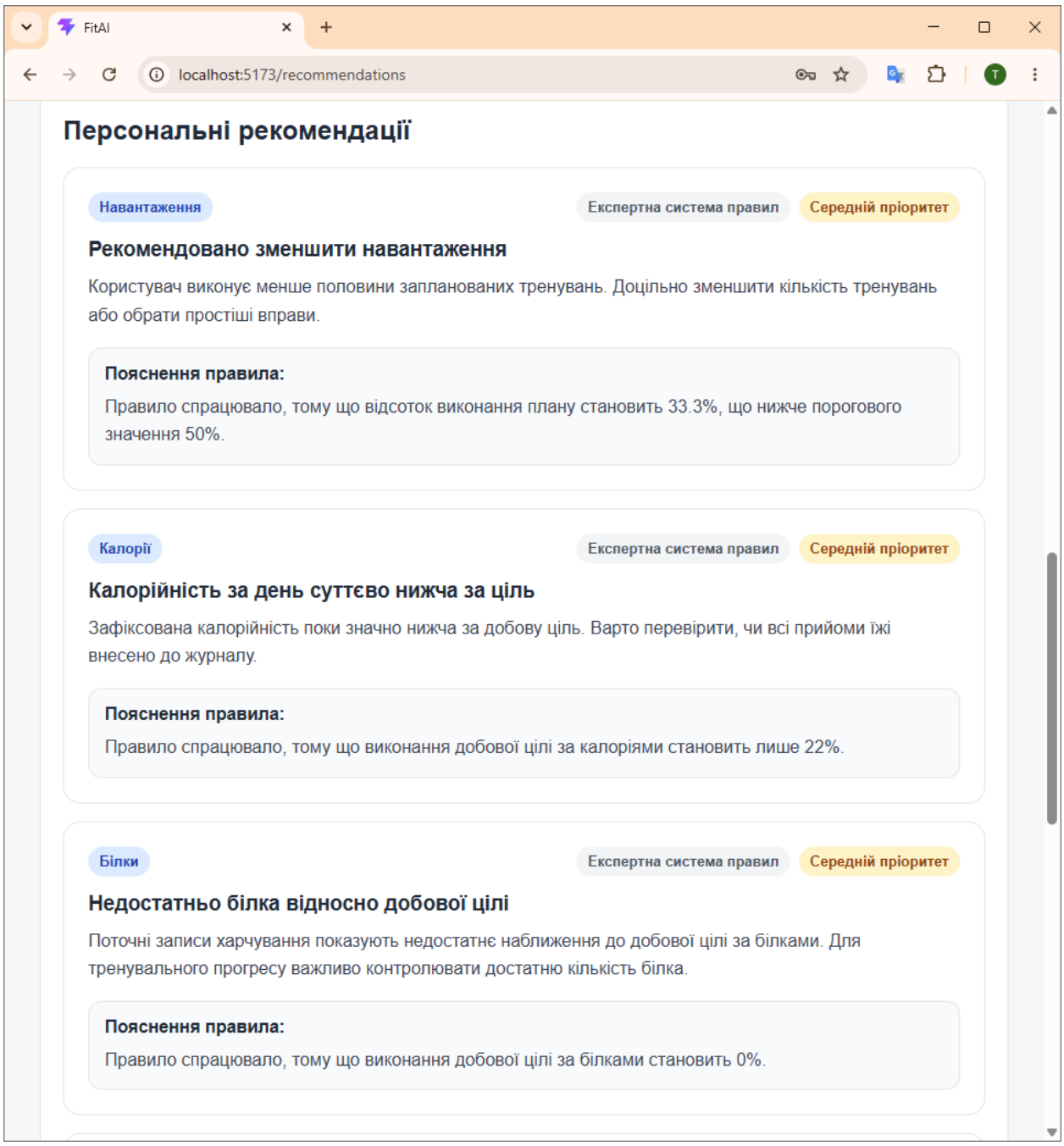

Рисунок Г.10 – Блок рекомендацій.

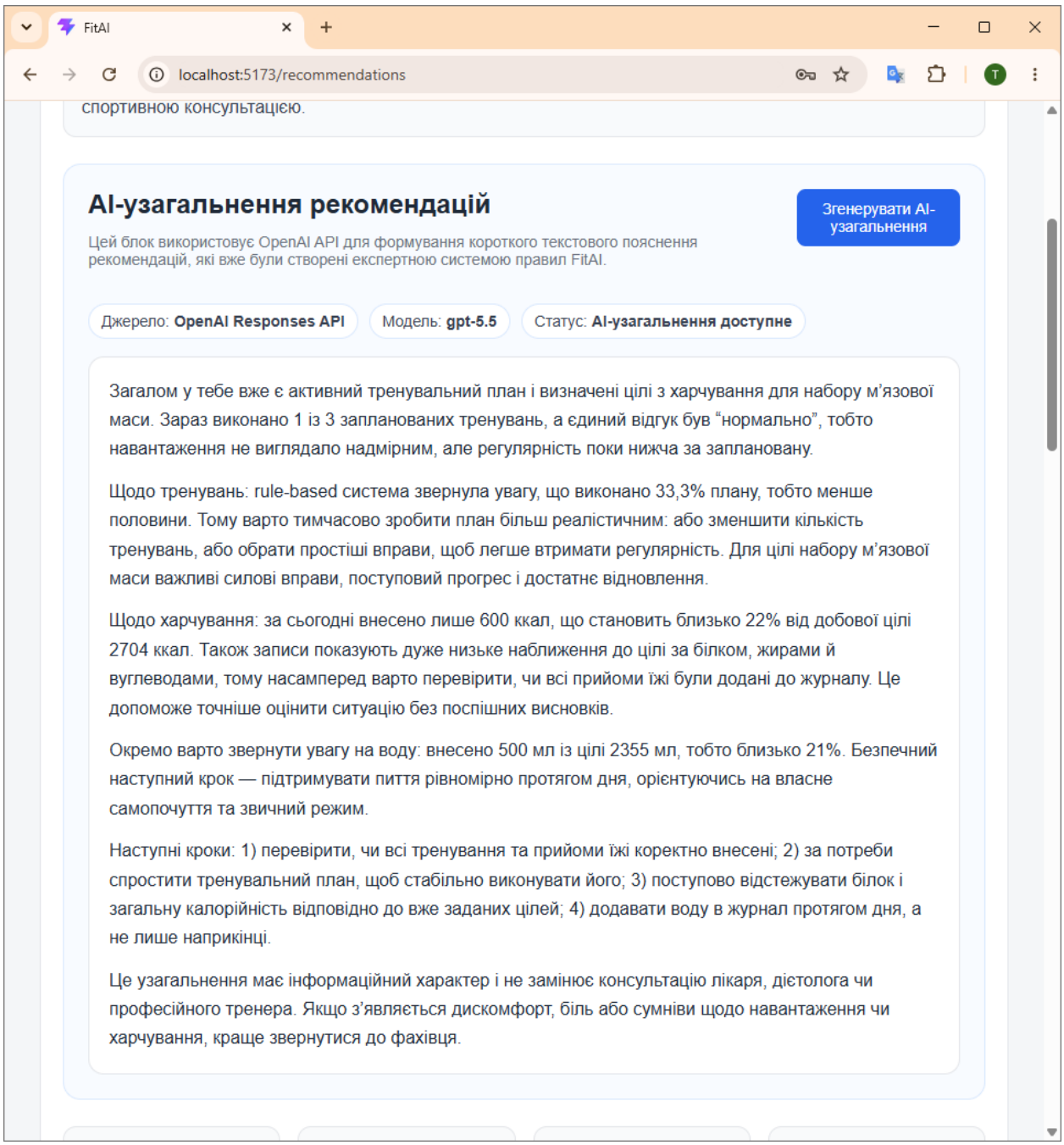

Рисунок Г.11 – Текстове AI-узагальнення рекомендацій.