Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Сервіс для створення персональної програми тренувань
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-3
Ярослав Литвин
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ас. Дмитро Малєєв
(посада, власне ім'я, прізвище)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Литвину Ярославу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Сервіс для створення персональної програми тренувань

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20 червня 20 25 р.

3. Вихідні дані до роботи Офіційні документації до OpenAI API, Firebase, React.js, TypeScript, Redux Toolkit, React Hook Form, Joy UI, Zod, Vite, Auth0, date-fns, Cloud Storage for Firebase та Google Firestore.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Проектування системи
3) Програмна реалізація

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Визначення проблеми	19.05.2025	виконано
3	Аналіз існуючих рішень	20.05.2025	виконано
4	Проектування компонентів системи	23.05.2025	виконано
5	Реалізація сервісу для тренувань	25.05.2025	виконано
6	Оформлення пояснювальної записки	03.06.2025	виконано
7	Захист	20.06.2025	виконано

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Дмитро Малєєв
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 100 с., 29 рис., 1 табл., 1 дод., 18 джерел.

ГЕНЕРАЦІЯ, ІНДИВІДУАЛЬНІСТЬ, ПЛАН, СЕРВІС, СПОРТ, ТРЕНУВАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, OPENAI API, REACT, REDUX, TYPESCRIPT.

Об’єктом дослідження є програмні засоби для персоналізованого планування тренувань.

Предметом дослідження є інтелектуальна система генерації фітнес-плану на основі індивідуальних параметрів користувача.

Метою роботи є розробка вебдодатку, який забезпечує формування персоналізованого тренувального плану із застосуванням генеративного штучного інтелекту.

Методи дослідження включають використання API OpenAI для генерації текстового контенту, засобів валідації даних, інтеграції з Firebase, а також сучасних підходів до розробки клієнтських інтерфейсів з використанням React.

У результаті роботи створено систему, яка автоматично формує персональний фітнес-план з урахуванням цілей, досвіду та фізичних характеристик користувача; новизна полягає в інтеграції генеративного ШІ у логіку побудови тренувань, а результати можуть бути використані у фітнес-додатках, системах здоров’я та персональних тренерах.

ABSTRACT

Bachelor’s thesis contains: 100 pp., 29 fig., 1 tabl., 1 ann., 18 references.

GENERATING, INDIVIDUALITY, PLAN, SERVICE, SPORT, TRAINING, ARTIFICIAL INTELLIGENCE, OPENAI API, REACT, REDUX, TYPESCRIPT

The object of the study is software for personalized workout planning.

The subject of the study is an intelligent fitness plan generation system based on individual user parameters.

The aim of this work is to develop a web application that provides personalized workout planning using generative artificial intelligence.

The research methods include the use of the OpenAI API for generating textual content, data validation tools, integration with Firebase, and modern frontend development techniques using React.

As a result, a system was developed that automatically creates personalized workout plans based on user goals, experience, and physical characteristics; the novelty lies in the integration of generative AI into the logic of workout generation, and the outcomes can be used in fitness applications, health systems, and personal coaching platforms.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – прикладний програмний інтерфейс;

HTTP – Hypertext Transfer Protocol – протокол передачі гіпертексту;

ID – Identifier – ідентифікатор;

REST – Representational State Transfer – передача репрезентативного стану;

UI – User Interface – інтерфейс користувача;

URL – Uniform Resource Locator – уніфікований вказівник ресурсу;

UX – User Experience – користувацький досвід.

ВСТУП

Сучасні тенденції у сфері охорони здоров’я, підвищення якості життя та розвитку цифрових технологій сприяли стрімкому зростанню інтересу до фітнесу як до важливого елементу повсякденної активності. З одного боку, дедалі більше людей прагнуть підтримувати належний рівень фізичної форми та піклуватися про власне здоров’я, а з іншого – зростає запит на зручні та адаптивні цифрові інструменти, які допомагають організувати процес тренувань без прив’язки до фітнес-клубів, розкладів занять або наявності професійного інструктора. У цьому контексті особливого значення набувають індивідуалізовані фітнес-сервіси, здатні забезпечити персоналізовані рекомендації та автоматизоване планування тренувань. Проблема полягає в тому, що більшість наявних рішень мають обмеження як у гнучкості, так і у доступності, особливо для користувачів, які мають специфічні потреби або обмежений досвід у використанні цифрових додатків.

Згідно з проведеним аналізом, сучасні вебдодатки для фітнесу умовно поділяються на два основні типи. Перший тип передбачає використання фіксованих тренувальних програм без суттєвої персоналізації. Другий – більш прогресивний – інтегрує можливості налаштування під потреби користувача, однак вимагає складної взаємодії, часто обмежується платним доступом або не підтримує повний цикл фітнес-супроводу, включаючи трекінг, візуалізацію результатів та підтримку носимих пристроїв. Крім того, далеко не всі сервіси використовують потенціал штучного інтелекту для адаптації тренувань у реальному часі або прогнозування фізичного прогресу. Це створює передумови для появи нового покоління фітнес-платформ, які поєднують автоматизацію, персоналізацію та простоту використання.

Актуальність теми дипломної роботи визначається саме потребою у створенні такого цифрового фітнес-сервісу, який би надавав користувачеві

можливість формувати персоналізовану програму тренувань з урахуванням фізичного стану, рівня підготовки, наявних обмежень та поставленої мети. Запропоноване рішення має забезпечити повний функціональний цикл – від створення профілю користувача та введення початкових параметрів до автоматичного формування тренувального плану, інтеграції з базою вправ, візуалізації інструкцій, ведення трекера прогресу та відображення розкладу занять. Враховуючи наявні технологічні можливості, зокрема сучасні вебфреймворки, хмарні сервіси, інструменти аналітики та API зі штучним інтелектом, реалізація такого сервісу є технічно досяжною та потенційно конкурентною на ринку цифрових рішень для підтримки здоров'я.

Цілю роботи є розробка вебдодатку, який дозволяє користувачам отримувати індивідуальні тренувальні плани на основі власних параметрів і цілей, супроводжені мультимедійними інструкціями, функціональністю трекінгу результатів та адаптивним розкладом. Запропонована система повинна поєднувати автоматичну генерацію контенту, зручну візуальну подачу інформації та простий інтерфейс. Проект орієнтований на користувачів, які прагнуть тренуватись вдома або поза межами спортивного залу, мають різний рівень підготовки, обмежений час або спеціальні потреби. Фітнес-сервіс має потенціал застосування не лише в особистому використанні, але й у сфері онлайн-консультування тренерами, інтеграції з корпоративними програмами підтримки здоров'я та освітніми ініціативами для популяризації активного способу життя.

У рамках даної кваліфікаційної роботи буде виконано аналіз предметної області, досліджено конкурентні рішення, окреслено потреби користувачів, сформовано функціональні та технічні вимоги до системи, а також спроектовано та реалізовано вебдодаток. Окрема увага буде приділена розробці логіки генерації персоналізованого плану тренувань, структуруванню бібліотеки вправ, впровадженню трекера прогресу та налаштуванню взаємодії з зовнішніми сервісами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз сучасного стану цифрових фітнес-сервісів

Сучасний фітнес стрімко трансформується під впливом цифрових технологій, перетворюючись з традиційної фізичної активності у багатофункціональну цифрову екосистему. В умовах глобального поширення інтернету та мобільних пристроїв фізична активність все частіше здійснюється за допомогою спеціалізованих програмних продуктів, які забезпечують персональну підтримку, контроль за прогресом, візуалізацію даних та гнучке налаштування під потреби користувача.

Цифрові фітнес-сервіси охоплюють широкий спектр функцій – від базових тренувальних планів до складних систем з елементами штучного інтелекту, що адаптують програму під індивідуальні характеристики. Широка доступність вебдодатків, інтеграція з носимими пристроями, можливість дистанційного супроводу та аналізу фізичної активності формують нові підходи до підтримки здорового способу життя. У цьому підрозділі розглянуто основні напрями розвитку фітнес-сервісів, їхні технологічні особливості та потенціал до персоналізації.

1.1.1 Роль фітнесу в житті сучасної людини

Фізична активність є одним з ключових чинників підтримки здоров’я людини у XXI столітті. Регулярні тренування сприяють зниженню ризику розвитку хронічних захворювань, покращують якість життя та підвищують загальну функціональність організму. Фітнес поступово перестає бути виключно спортивною активністю й перетворюється на обов’язкову складову способу життя сучасної людини, яка прагне зберегти здоров’я, енергію та психоемоційну стабільність.

Фізичні навантаження позитивно впливають на роботу серцево-судинної системи, органів дихання, опорно-рухового апарату, а також сприяють нормалізації артеріального тиску та рівня глюкози в крові. Крім того, активність покращує когнітивні функції, сприяє підвищенню концентрації уваги, якості сну і загального психоемоційного стану. В умовах цифрової епохи ці переваги все частіше досягаються за допомогою спеціалізованих сервісів.

Вплив фізичних вправ на різні органи та системи організму добре ілюструє узагальнена схема ефектів фізичної активності (рисунок 1.1). На ній показано, як регулярне тренування зменшує ризики серцево-судинних, онкологічних, респіраторних захворювань, покращує роботу печінки, нирок, підшлункової залози та кісткової тканини. Також фізична активність сприяє зростанню рівня гемоглобіну та лімфоцитів у крові, що безпосередньо впливає на імунну систему.

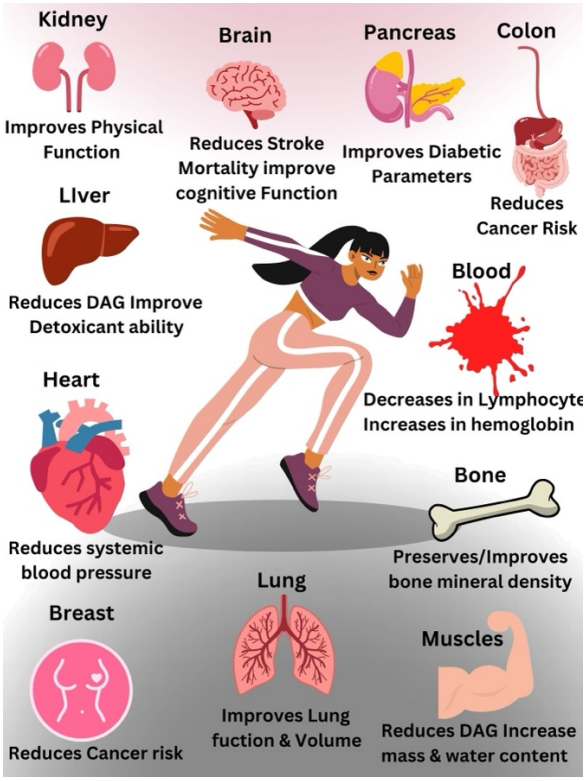

Рисунок 1.1 – Користь від спорту для організму людини

У контексті здорового способу життя фітнес не лише виконує функцію підтримки фізичної форми, а й відіграє роль інструменту профілактики, самоконтролю та управління власним тілом. Саме тому актуальність розвитку цифрових рішень у сфері фітнесу зростає, адже вони дають можливість значно ефективніше реалізовувати оздоровчі практики завдяки персоналізованому підходу, наочній аналітиці та постійному зворотному зв’язку.

1.1.2 Вплив цифрових технологій на розвиток фітнес-індустрії

Цифрові технології значно змінили підхід до організації фітнес-процесів, надавши користувачам нові можливості для ефективного контролю та управління фізичною активністю.

Завдяки розвитку мобільних додатків, хмарних сервісів та інтеграції з носимими пристроями, тренування стали доступними у будь-якому місці й у будь-який час. Користувачі отримали змогу персоналізувати свої програми, вести трекінг результатів, переглядати відеоінструкції та отримувати рекомендації в реальному часі.

Однією з найпомітніших тенденцій останніх років є злиття фітнесу з екосистемами цифрових пристроїв. Смартфони, планшети, смартгодинники та телевізори синхронізуються між собою, забезпечуючи комплексну взаємодію користувача з цифровим тренером.

Це дозволяє автоматично фіксувати активність, оцінювати навантаження, нагадувати про тренування та адаптувати їх до змін у фізичному стані. Такий підхід істотно підвищує мотивацію та сприяє систематичному дотриманню тренувального режиму. Приклад реалізації подібної інтеграції можна побачити нижче (рисунок 1.2), де зображено взаємодію кількох пристроїв з єдиною фітнес-платформою.

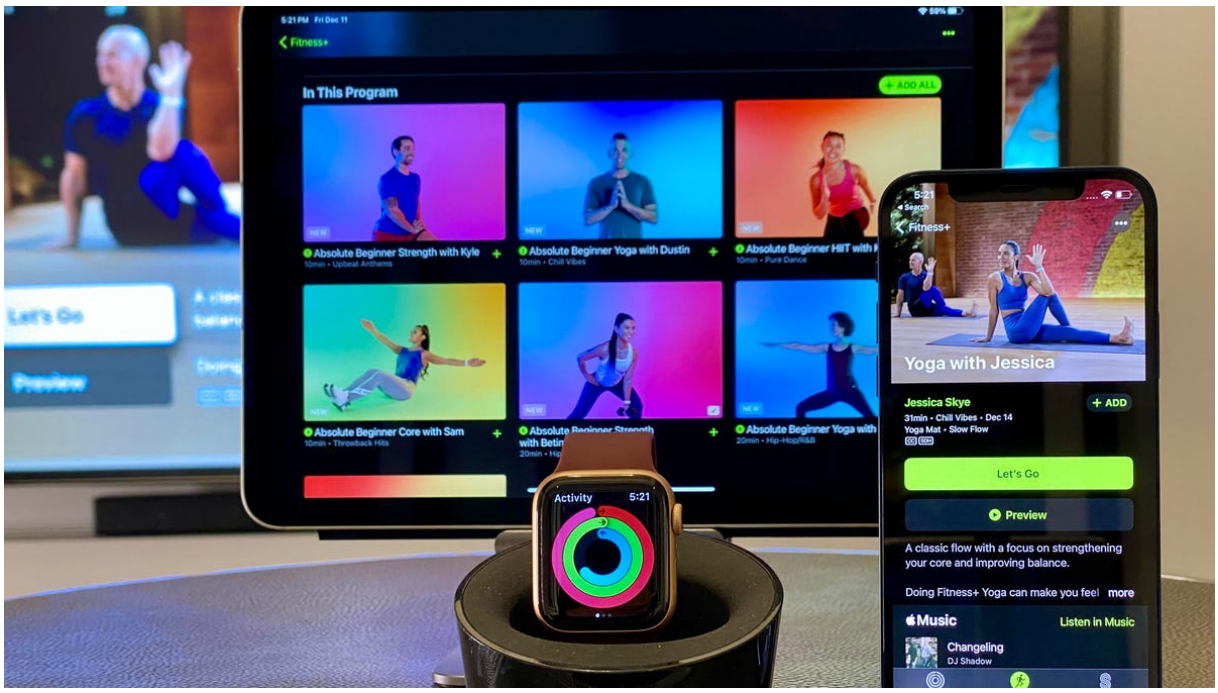

Рисунок 1.2 – Використання кількох синхронізованих цифрових пристроїв для взаємодії з тренувальною платформою

Користувач може обрати тренування на планшеті або телевізорі, водночас отримуючи підказки та зворотний зв’язок на смартгодиннику, а також мати доступ до інформації через мобільний додаток.

Це свідчить про високий рівень технічного розвитку фітнес-сервісів та їхню здатність забезпечувати персоналізовану та зручну взаємодію з користувачем.

1.1.3 Особливості вебдодатків для фітнесу та здорового способу життя

Сучасні вебдодатки у сфері фітнесу значно розширили свої функціональні можливості, перетворившись на повноцінні цифрові платформи для підтримки здорового способу життя. Вони дозволяють не лише виконувати тренування за інструкціями, а й аналізувати прогрес, отримувати персоналізовані поради, інтегрувати дані з носимих пристроїв та ділитися результатами в соціальних мережах. Такі сервіси орієнтовані на

максимальну зручність, індивідуалізацію та залучення користувача до регулярної активності.

Далі можна побачити перелік функціональних можливостей, які зазвичай реалізовані у сучасних фітнес-додатках (рисунок 1.3). Серед базових функцій – створення профілю користувача, тренувальні й харчові плани, трекінг активності, система досягнень і підтримка. До розширених належать: доступ до онлайн-тренерів, прямі трансляції, інтеграція з носимими пристроями, підтримка Google Fit та Apple Health, а також синхронізація з соціальними мережами.

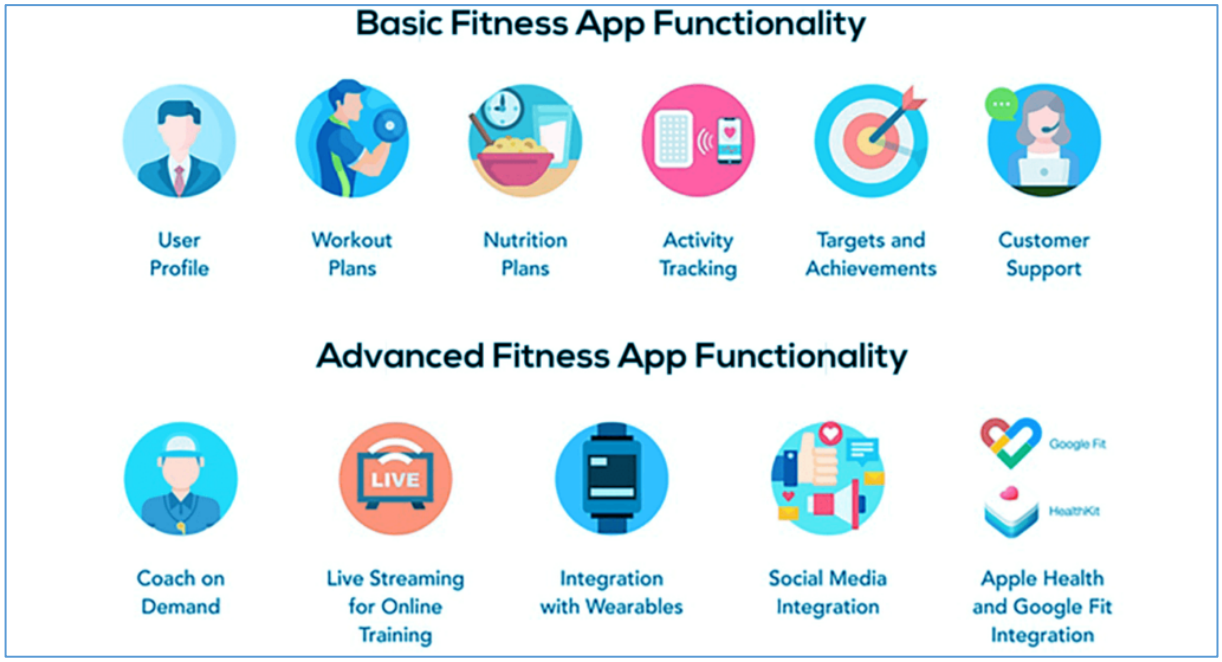

Рисунок 1.3 – Базові та розширені функціональні можливості сучасних фітнес-додатків

Більшість популярних фітнес-сервісів реалізують гнучке налаштування під цілі та фізичні можливості користувача. Наприклад, можна обрати рівень підготовки, бажаний тип навантаження (аеробне, силове, змішане) і тривалість занять. При цьому користувач отримує доступ

до мультимедійного контенту з інструкціями, що підвищує ефективність виконання вправ і мінімізує ризик травм.

Такі додатки також забезпечують мотиваційний компонент – система досягнень, візуалізація результатів і рекомендації щодо подальших кроків допомагають підтримувати регулярність занять. Окремі платформи дозволяють встановлювати персональні цілі, які поступово адаптуються під прогрес користувача, створюючи ефект супроводу реальним тренером.

Усе це свідчить про те, що вебдодатки у сфері фітнесу вже давно вийшли за межі простих відеоуроків. Вони стали комплексними інструментами для управління фізичною активністю, що поєднують у собі аналітику, персоналізацію, гейміфікацію та соціальну взаємодію. Саме тому такі сервіси є важливою складовою цифрової трансформації здорового способу життя.

1.1.4 Персоналізація тренувальних програм у сучасному цифровому середовищі

Персоналізація тренувального процесу стала однією з головних переваг цифрових фітнес-сервісів. Сучасні вебдодатки дозволяють користувачам обирати типи вправ, рівень складності, тривалість занять і навіть режим навантаження відповідно до індивідуальних цілей, таких як схуднення, набір м’язової маси чи підтримка загального тонусу. Такий підхід забезпечує більш високу ефективність тренувань та сприяє підвищенню мотивації, оскільки програма враховує фізичний стан, вік, стать та наявність обмежень.

Цифрові платформи застосовують різні методи для досягнення персоналізації – від простих опитувальників до складних алгоритмів адаптації контенту. Дані, що вводить користувач на етапі реєстрації або під час моніторингу прогресу, стають основою для динамічного формування індивідуального плану. Таким чином, фітнес-додатки перетворюються на

інструмент постійного супроводу, який не лише навчає, а й підлаштовується під зміни в стані користувача та забезпечує безперервний зворотний зв’язок.

1.1.5 Тренди розвитку фітнес-сервісів

Сучасні фітнес-сервіси продовжують активно розвиватися, охоплюючи все нові напрями та технології. До стандартного функціоналу вже давно додалися можливості, пов’язані з доповненою і віртуальною реальністю, мобільними додатками для носимих пристроїв, трекерами прогресу, інтеграцією з медичними платформами та застосуванням штучного інтелекту. Далі можна побачити перелік актуальних трендів, що визначають розвиток фітнес-додатків у цифрову епоху (рисунок 1.4).

Рисунок 1.4 – Тренди у фітнес-сервісів

У цьому переліку представлені як традиційні елементи – домашні вправи, персональні тренери, навчальні відео та плани тренувань – так і новітні цифрові рішення, що підвищують ефективність і занурення

користувача. Такі інновації як програми з доповненою реальністю або додатки для health clubs надають нові можливості для залучення аудиторії. Водночас фокус на індивідуальний підхід з використанням штучного інтелекту та автоматизованих рекомендацій відкриває перспективи ще глибшої персоналізації тренувального процесу.

1.1.6 Штучний інтелект як рушійна сила персоналізації фітнес-програм

Застосування штучного інтелекту у фітнес-індустрії стало новим етапом у розвитку персоналізованих цифрових рішень. Алгоритми машинного навчання здатні аналізувати великі обсяги даних користувача, виявляти тренди та формувати рекомендації, які підходять конкретній людині з урахуванням її мети, фізичних параметрів та історії активності. Це дозволяє вийти за межі статичних програм тренувань і зробити процес динамічним, адаптивним і максимально ефективним.

Далі можна побачити основні напрями використання штучного інтелекту у фітнес-сфері, які сьогодні активно впроваджуються в цифрові сервіси (рисунок 1.5). Серед них – інтелектуальні тренажери, персоналізовані плани на основі AI, аналітика біометричних даних, віртуальні заняття, аналіз тіла в реальному часі, а також носимі пристрої з функціями прогнозування навантаження. Такі рішення підвищують точність рекомендацій і роблять взаємодію з додатком схожою на консультування з живим фахівцем.

Інтеграція AI-моделей у мобільні додатки дозволяє автоматично коригувати програму тренувань на основі фактичної продуктивності та фізіологічних змін користувача. Наприклад, якщо зафіксоване зниження рівня активності або перевтома, система може запропонувати зміну навантаження або альтернативний вид активності. Також активно

використовуються системи розпізнавання рухів, які оцінюють техніку виконання вправ і надають підказки для її покращення.

Рисунок 1.5 – Використання ШІ у фітнес індустрії

Таким чином, штучний інтелект у фітнес-сервісах виконує не лише аналітичну, але й супервізійну функцію – постійно адаптуючи програму до змін у фізичному стані користувача. Це відкриває нові горизонти для формування гнучких, індивідуальних тренувальних маршрутів, де кожне рішення приймається на основі даних, а не загальних шаблонів. У поєднанні з інтерфейсами віртуальної та доповненої реальності такі сервіси формують нову якість тренувального досвіду.

1.2 Дослідження конкурентного середовища у сфері персоналізованих тренувань

У сучасному цифровому середовищі ринок фітнес-додатків представлений великою кількістю конкурентних рішень, які надають користувачам широкий спектр функцій для організації тренувального

процесу. Ці сервіси орієнтовані на різні цільові аудиторії, пропонуючи як універсальні рішення для масового користувача, так і спеціалізовані платформи для професійного тренінгу чи відновлення після травм. Вони відрізняються рівнем персоналізації, дизайном інтерфейсу, кількістю доступного контенту та підтримкою інтеграцій із носимими пристроями.

Аналіз конкурентного середовища дозволяє виявити сильні та слабкі сторони існуючих продуктів, оцінити рівень інноваційності та зручності для користувача, а також зрозуміти, які функції є стандартом, а які залишаються унікальними. Саме тому порівняння функціональних можливостей, підходів до взаємодії з користувачем та моделей персоналізації є необхідним етапом при проєктуванні власного фітнес-сервісу. У підрозділах цього розділу буде здійснено детальний аналіз ринку та визначено напрями для вдосконалення.

1.2.1 Критерії оцінювання конкурентних сервісів

Для об’єктивного аналізу конкурентних фітнес-сервісів необхідно визначити чіткі критерії оцінювання, які дозволяють порівнювати платформи за однаковими параметрами. Такий підхід забезпечує узагальнену картину ринку та виявляє найбільш важливі особливості реалізації функціоналу, орієнтованого на кінцевого користувача. Зазвичай критерії поділяються на функціональні, технічні та користувацькі.

До функціональних критеріїв належать наявність індивідуальних тренувальних планів, гнучкість налаштувань, трекінг прогресу, підтримка мультимедійного контенту, рекомендації на основі даних та інші можливості персоналізації. Оцінюється також широта бази вправ, можливість інтеграції з іншими сервісами, наявність онлайн-тренерів або автоматичних підказок під час тренування.

Користувацькі критерії охоплюють зручність інтерфейсу, простоту навігації, адаптивність до різних пристроїв, швидкодію, стабільність роботи та рівень задоволеності користувачів. Додатково можуть враховуватись такі

аспекти, як дизайн, доступність мовної локалізації, політика конфіденційності та вартість підписки. Сукупність цих показників дозволяє комплексно оцінити, наскільки конкурентоспроможним є той чи інший фітнес-додаток.

1.2.2 Аналіз існуючих вебплатформ для створення тренувальних програм

На ринку фітнес-додатків існує велика кількість вебплатформ, що пропонують створення індивідуальних тренувальних програм. Серед найпоширеніших варто відзначити такі сервіси як Freeletics, Nike Training Club, FitOn, Jefit та MyFitnessPal. Кожен із них має свій підхід до побудови тренувального процесу, набір функцій та орієнтацію на певні категорії користувачів.

Freeletics вирізняється інтенсивними персоналізованими тренуваннями без обладнання, які адаптуються до результатів користувача після кожного заняття.

Nike Training Club пропонує широку бібліотеку безкоштовних тренувань з високоякісним відеоконтентом, а також спеціальні програми, розроблені разом із професійними тренерами.

FitOn фокусується на доступності та зручності: всі тренування безкоштовні, а інтеграція із календарем дозволяє краще планувати заняття.

Jefit більше орієнтований на силові тренування та бодібілдинг. Він містить детальну базу вправ, можливість фіксувати підходи, вагу та повторення, а також має планувальник тренувань.

MyFitnessPal, хоча спершу позиціонувався як сервіс для підрахунку калорій, згодом інтегрував базовий функціонал тренувань і став прикладом комплексного підходу до здоров’я – поєднуючи харчування, активність і аналітику.

Ці платформи є гарною відправною точкою для вивчення ринку, однак жодна з них не пропонує одночасно повну персоналізацію, адаптивність, простоту інтерфейсу та глибоку інтеграцію з трекерами. Це створює простір для нових рішень, які могли б поєднати найкращі риси розглянутих сервісів.

1.2.3 Порівняльний аналіз функціональних можливостей прямих конкурентів

Порівняльний аналіз функціональних можливостей прямих конкурентів дозволяє систематизувати доступні рішення на ринку фітнес-сервісів за ключовими параметрами. Такий підхід допомагає виявити, які саме функції найчастіше реалізуються у сучасних вебдодатках, а також які з них стали галузевим стандартом. Для аналізу було відібрано п’ять популярних платформ: Freeletics, Nike Training Club, FitOn, Jefit та MyFitnessPal.

Основними критеріями для порівняння стали: наявність персоналізованих тренувальних планів, підтримка відеоінструкцій, інтеграція з носимими пристроями, функціонал відстеження прогресу, синхронізація з календарем, можливість налаштування цілей, а також додаткові функції на кшталт моніторингу харчування або аналітики результатів. Усі ці показники дозволяють оцінити рівень комплексності та гнучкості кожного сервісу.

Зведені результати аналізу представлено у таблиці 1.1, що містить інформацію про підтримку вищезазначених функцій у кожному з розглянутих сервісів. Це дає змогу швидко зіставити платформи між собою за конкретними характеристиками. На основі отриманих даних у наступному підрозділі буде зроблено висновки щодо переваг і обмежень кожного рішення з позиції користувача.

Таблиця 1.1 – Порівняльний аналіз функціональних можливостей конкурентів

Функціональність	Freeletics	Nike Club	FitOn	Jefit	MyFitnessPal
Персоналізовані тренування	Є	Є	Немає	Є	Немає
Відеоінструкції	Є	Є	Є	Є	Немає
Відстеження прогресу	Є	Є	Є	Є	Є
Моніторинг харчування	Немає	Немає	Немає	Немає	Є
Гейміфікація	Є	Є	Є	Немає	Є

1.2.4 Виявлення сильних та слабких сторін існуючих рішень

Проведений аналіз дозволяє виявити основні сильні та слабкі сторони існуючих фітнес-платформ, орієнтованих на створення персоналізованих тренувальних програм. Кожен із розглянутих сервісів має певні переваги, однак також демонструє обмеження, які впливають на повноту реалізації запитів користувачів.

Freeletics має сильну сторону у вигляді високого рівня персоналізації тренувань і адаптації плану після кожного заняття. Проте його слабким місцем є відсутність безкоштовного доступу до повного функціоналу, що обмежує привабливість для широкої аудиторії. Крім того, сервіс недостатньо інтегрований із календарем та не пропонує можливості моніторингу харчування.

Nike Training Club пропонує велику базу безкоштовних тренувань, що є його беззаперечною перевагою. Інтерфейс додатку простий і доступний навіть для новачків. Однак слабкою стороною є відсутність глибокої персоналізації тренувальних планів та обмеження в гнучкому налаштуванні індивідуальних програм.

FitOn вирізняється тим, що пропонує безкоштовний доступ до всього контенту, включаючи відеоінструкції з тренерами. Сервіс має інтеграцію з календарем та підтримку носимих пристроїв. Слабкою стороною є відсутність повноцінної персоналізації тренувань і нестача більш розширених аналітичних функцій для відстеження фізичного прогресу.

Jefit має розвинену базу вправ для силових тренувань і потужні можливості для трекінгу підходів та результатів. Серед недоліків варто зазначити відсутність безкоштовного доступу до повного функціоналу, а також недостатню орієнтацію на користувачів, які цікавляться іншими типами фізичної активності, окрім силових навантажень.

MyFitnessPal є прикладом комплексного рішення, яке поєднує тренування з моніторингом харчування. Його перевага полягає у гнучкості налаштувань цілей і аналітиці даних. Проте тренувальні функції залишаються допоміжними, а основний акцент сервісу робиться саме на харчуванні, що обмежує його сприйняття як повноцінної тренувальної платформи.

Таким чином, жоден із аналізованих сервісів не охоплює повний спектр необхідних можливостей для створення персоналізованої фітнес-програми з високим рівнем адаптивності та комплексної підтримки здоров’я. Це визначає актуальність розробки нового рішення, що поєднуватиме найкращі властивості існуючих платформ та усуватиме їхні недоліки.

1.2.5 Визначення можливостей для вдосконалення на основі аналізу конкурентів

Аналіз функціональних можливостей існуючих фітнес-сервісів показав, що на ринку залишається низка незадоволених потреб користувачів. Незважаючи на широку пропозицію, жоден із досліджених продуктів не забезпечує одночасно глибоку персоналізацію тренувальних

програм, комплексне відстеження фізичного стану та зручний безкоштовний доступ до основного функціоналу. Це відкриває можливості для розробки нового рішення, яке буде сфокусоване на максимальній адаптивності під індивідуальні характеристики користувача та простоті у використанні.

Крім того, перспективним напрямом удосконалення є інтеграція функціоналу прогнозування прогресу, динамічної адаптації навантаження та розширеної аналітики показників фізичної активності. Значну роль може відігравати також підтримка мотиваційних механізмів, таких як гейміфікація, персональні рекомендації та системи віртуальних досягнень. У підрозділах буде більш детально окреслено, які саме можливості планується реалізувати на основі виявлених недоліків конкурентних рішень.

1.3 Аналіз потреб і очікувань користувачів фітнес-сервісу

Ефективність фітнес-додатків значною мірою залежить від того, наскільки точно вони відповідають реальним потребам та очікуванням користувачів. Сучасні користувачі очікують не просто універсальний тренувальний інструмент, а індивідуалізовану платформу, яка враховує їхні цілі, рівень фізичної підготовки, стан здоров’я та спосіб життя. Саме тому вивчення цільової аудиторії та її вимог є важливим етапом проєктування функціональних можливостей майбутнього сервісу.

Аналіз потреб користувачів дозволяє визначити критично важливі функції, очікування щодо зручності інтерфейсу, рівня мотивації та способів зворотного зв’язку. Різні категорії користувачів мають власні пріоритети – одні потребують максимально простого керування тренуваннями, інші шукають повну аналітику та прогнози, побудовані на основі власних даних. У наступних підрозділах розглянуто ключові характеристики цільової аудиторії, її функціональні запити.

1.3.1 Портрет цільової аудиторії

Цільова аудиторія фітнес-сервісу охоплює широку групу користувачів, які прагнуть покращити свій фізичний стан, підтримувати здоров’я або досягати конкретних спортивних цілей. Серед них можна виділити людей різного віку та рівня фізичної підготовки – від початківців до просунутих спортсменів. Також аудиторія включає користувачів з обмеженим часом для занять, які надають перевагу тренуванням вдома або у зручний для себе час без прив’язки до спортивних залів.

Важливою особливістю сучасної аудиторії є прагнення до персоналізованого підходу та простоти використання сервісу. Користувачі очікують, що програма буде адаптуватися під їхні фізичні можливості, цілі та спосіб життя. Зростає також попит на цифрові рішення, які надають чіткі рекомендації, візуалізацію прогресу та інтеграцію з іншими платформами для моніторингу здоров’я. Ці характеристики визначають базові очікування, які необхідно враховувати при проектуванні сервісу.

1.3.2 Основні вимоги користувачів до функціоналу сервісу

Основні вимоги користувачів до функціоналу фітнес-сервісу зводяться до зручності, доступності та ефективності тренувального процесу. Користувачі очікують, що платформа буде простою у використанні, матиме інтуїтивно зрозумілий інтерфейс та дозволить швидко налаштовувати параметри тренувань відповідно до індивідуальних цілей. Важливою умовою є також наявність готових рішень для різних рівнів фізичної підготовки та можливість коригування програми в залежності від змін у стані користувача.

Ще однією важливою вимогою є надання якісного мультимедійного контенту – відеоінструкцій, описів вправ та рекомендацій щодо техніки виконання. Користувачі очікують можливості контролювати свій прогрес

через аналітику, трекінг виконаних занять і моніторинг основних показників фізичного стану. Підтримка носимих пристроїв, можливість інтеграції з календарем та нагадування про тренування також входять до числа функцій, які підвищують цінність сервісу для кінцевого споживача.

1.3.3 Потреба у персоналізації тренувань та відстеженні прогресу

Персоналізація тренувань є однією з ключових потреб сучасних користувачів фітнес-сервісів. Більшість людей прагнуть отримувати індивідуальні рекомендації, що враховують їхній рівень підготовки, фізичні особливості, цілі та обмеження у здоров'ї. Користувачі очікують, що сервіс самостійно буде адаптувати навантаження, підбирати оптимальні типи вправ і пропонувати зміну програми відповідно до прогресу або змін у способі життя.

Важливою складовою персоналізації є також можливість відстеження власного прогресу. Користувачі хочуть бачити результати своєї роботи у вигляді графіків, статистики виконаних тренувань, змін ваги або інших показників фізичного стану. Такий функціонал не тільки допомагає об'єктивно оцінити досягнення, але й підвищує мотивацію до регулярних занять і подальшого розвитку.

1.3.4 Очікування користувачів щодо інтерфейсу та зручності використання

Очікування користувачів щодо інтерфейсу фітнес-сервісу зводяться до вимог простоти, зрозумілості та інтуїтивної навігації. Вебдодаток має забезпечувати швидкий доступ до основних функцій, мінімізувати кількість зайвих дій та не перевантажувати користувача складними меню або заплутаними налаштуваннями. Особливе значення має адаптивність

інтерфейсу до різних типів пристроїв, оскільки більшість користувачів працюють з мобільних телефонів або планшетів.

Зовнішній вигляд додатку також відіграє важливу роль у сприйнятті сервісу. Користувачі віддають перевагу сучасному дизайну із приємною кольоровою палітрою, чіткою структурою елементів та логічним розташуванням функціональних блоків. Наявність візуальних підказок, прогрес-барів, інтеграція мультимедійних матеріалів та швидке реагування інтерфейсу на дії користувача суттєво підвищують рівень задоволеності від взаємодії з додатком.

1.3.5 Мотиваційні фактори використання фітнес-сервісів

Мотиваційні фактори відіграють важливу роль у залученні та утриманні користувачів фітнес-сервісу. Користувачі прагнуть бачити прогрес, отримувати визнання своїх досягнень і мати постійний стимул для продовження тренувань.

Серед основних бар’єрів використання фітнес-додатків можна виділити втрату інтересу через одноманітність тренувань, складність адаптації програм до змін у фізичному стані та відсутність оперативного зворотного зв’язку. Подолання цих бар’єрів можливе за рахунок впровадження адаптивних тренувальних планів, різноманітності контенту, нагадувань і підтримки персоналізованих рекомендацій, що дозволяє зберігати мотивацію користувача протягом тривалого часу.

1.4 Функціональні можливості сервісу для створення персональної програми тренувань

Розробка ефективного фітнес-сервісу потребує чіткого визначення функціональних можливостей, які забезпечать реалізацію основних сценаріїв користувача. Вебдодаток повинен не лише містити базові

інструменти для створення тренувальної програми, але й бути достатньо гнучким, щоб адаптуватися під індивідуальні параметри користувача, забезпечити зручність взаємодії та підтримку трекінгу результатів. Формування структури функцій має відбуватись з урахуванням як потреб цільової аудиторії, так і технічних можливостей реалізації.

Крім функціональних аспектів, важливо також окреслити технічні вимоги до вебдодатку, які забезпечать його стабільність, безпеку та масштабованість. Йдеться про вимоги до збереження даних, інтеграції з іншими платформами, підтримки адаптивного інтерфейсу та можливості розширення функціоналу в майбутньому. У наступних підрозділах буде представлено загальну функціональну модель сервісу та окреслено ключові технічні параметри, що впливають на архітектуру рішення.

1.4.1 Опис ключових функціональних можливостей

Основні функціональні можливості вебдодатку визначаються його здатністю повністю покривати основні сценарії використання, що є актуальними для цільової аудиторії. Запропонований фітнес-сервіс має включати набір інструментів, які забезпечують персоналізацію тренувального процесу, контроль за прогресом та зручну взаємодію користувача з платформою. Кожна функція є логічною складовою єдиної системи підтримки фізичної активності.

Далі можна побачити візуалізацію ключових функцій, передбачених у структурі додатку (рисунок 1.6). До них належать: реєстрація та автентифікація користувача, доступ до бібліотеки вправ, трекер прогресу, автоматична генерація тренувальної програми, відео- та текстові інструкції до вправ, а також розклад тренувань. Такий набір дозволяє реалізувати повний цикл користувацької взаємодії – від старту до постійного самоконтролю і корекції.

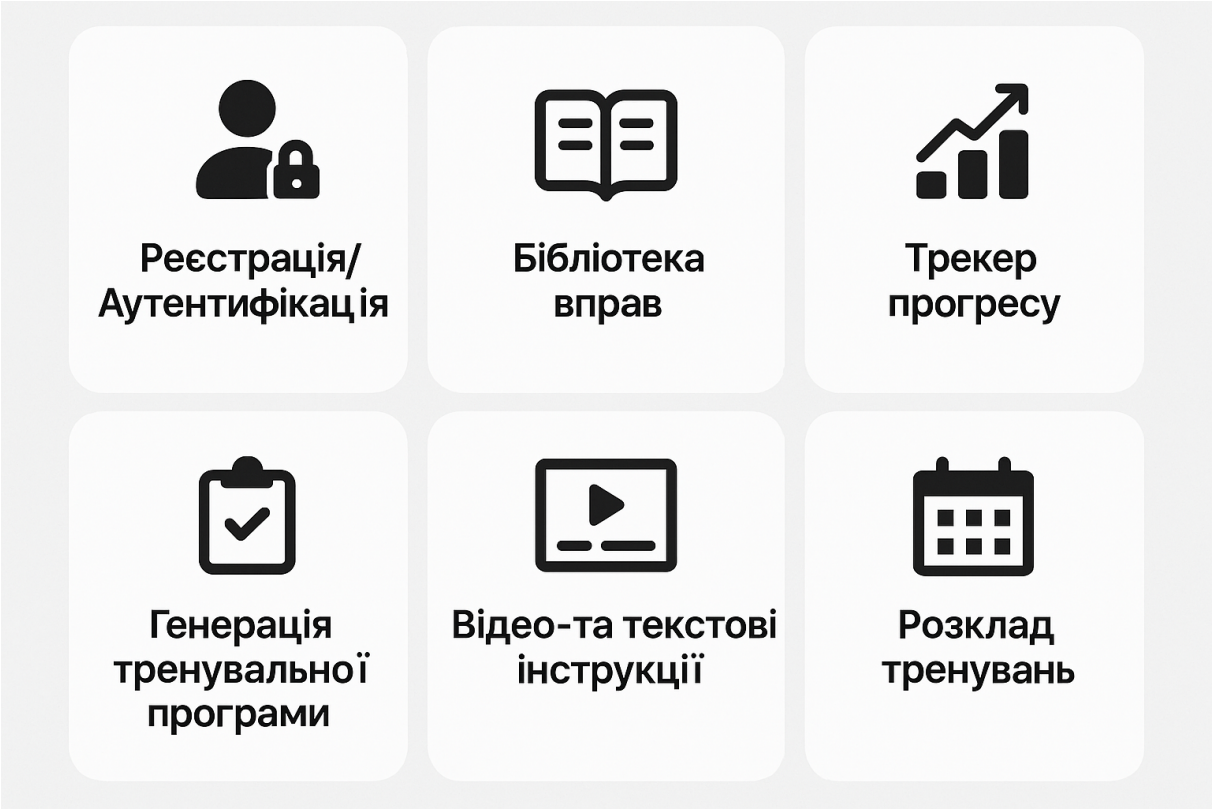

Рисунок 1.6 – Функціонал сервісу

Функція реєстрації та автентифікації забезпечує створення персонального облікового запису з можливістю збереження індивідуальних налаштувань, результатів тренувань та цілей. Бібліотека вправ містить каталог вправ, розділених за типами навантаження, групами м’язів і рівнем складності, що дозволяє користувачу ознайомитися з технікою виконання і самостійно формувати уявлення про структуру занять.

Генерація тренувальної програми базується на введених параметрах користувача – цілі, рівень фізичної підготовки, обмеження та уподобання. Алгоритм створює план на тиждень або місяць, автоматично розподіляючи вправи за днями і враховуючи баланс навантаження. Додатково до кожної вправи додаються відео- та текстові інструкції, які допомагають правильно виконувати завдання без потреби в зовнішньому супроводі.

Трекер прогресу та розклад тренувань забезпечують зручне планування і контроль за дотриманням програми. Користувач має змогу

фіксувати виконані заняття, відмічати прогрес у вазі, кількості повторень або часу активності. Інтерфейс календаря дозволяє бачити заплановані тренування і швидко реагувати на зміни в режимі, що підвищує гнучкість використання сервісу.

1.4.2 Визначення технічних вимог та обмежень

Для реалізації вебдодатку було визначено набір технічних рішень, які забезпечують сучасність, масштабованість та швидкодію системи. На стороні клієнта застосовуються технології з акцентом на високу інтерактивність і зручність користувача. Інтерфейс побудовано з використанням React, а для стилізації компонентів обрано бібліотеку Joy UI, що дозволяє створювати естетичний та адаптивний дизайн. Для маршрутизації між сторінками використовується react-router, а взаємодія з формами реалізована через react-hook-form. Авторизація організована на основі зовнішнього постачальника, що гарантує безпечний вхід до системи. Весь клієнтський код написано з використанням TypeScript, що підвищує стабільність розробки, а середовище збирання реалізовано за допомогою Vite, який забезпечує швидкий час компіляції.

На серверній стороні застосовано фреймворк Nest.js, який дозволяє створювати структурований і розширюваний сервер на базі TypeScript. Дані зберігаються у хмарній базі Google Firestore, що є частиною Firebase і добре масштабується для реального часу. Додатково використовується Google Cloud Bucket для зберігання файлів та OpenAI API для генерації інтелектуального контенту. Така архітектура дозволяє гнучко інтегрувати нові функції й адаптувати сервіс до змінних потреб користувачів.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Аналіз вимог розроблюваної системи

Проектування системи розпочинається з етапу аналізу вимог, який відіграє ключову роль у формуванні технічного бачення майбутнього програмного продукту. На цьому етапі визначаються загальні технічні, функціональні та інтерфейсні вимоги до системи, які дозволяють побудувати ефективну архітектуру і закласти основу для реалізації всіх ключових можливостей вебдодатку.

Аналіз вимог передбачає вивчення очікувань цільових користувачів, особливостей функціонального призначення системи та умов її роботи у вебсередовищі. Визначення чітких і досяжних вимог дозволяє зменшити ризики на наступних етапах розробки, підвищити якість проєкту та забезпечити відповідність створеного сервісу поставленим завданням.

У рамках цього підрозділу будуть розглянуті технічні обмеження середовища виконання, потреби в обчислювальних ресурсах, підтримка масштабованості та безпеки, а також вимоги до функціонального наповнення й зручності користування. Такий підхід забезпечить цілісне розуміння логіки роботи майбутнього фітнес-сервісу, що є основою для успішної реалізації програмної системи.

2.1.1 Аналіз технічних вимог сервісу

Розробка вебдодатку потребує врахування низки технічних вимог, які забезпечують його стабільну роботу, масштабованість і безпеку. До таких вимог належать вибір відповідної архітектури, підтримка адаптивності інтерфейсу, забезпечення високої продуктивності при зростанні кількості користувачів та інтеграція з хмарними сервісами. Важливим також є розподіл навантаження між клієнтською та серверною частинами системи.

З урахуванням того, що додаток має працювати у вебсередовищі, він повинен бути сумісним з основними браузерами, підтримувати роботу на різних типах пристроїв і мати мінімальні вимоги до обчислювальних ресурсів користувача. Реалізація клієнтської частини передбачає використання сучасних технологій, які дозволяють забезпечити швидке завантаження сторінок, ефективну обробку даних та асинхронну взаємодію з сервером.

З боку серверної частини технічні вимоги включають надійну обробку запитів, збереження та оновлення даних у базі, а також захист від несанкціонованого доступу. Важливо передбачити підтримку хмарного розгортання, щоб мати змогу масштабувати систему відповідно до навантаження. Використання контейнеризації забезпечує гнучкість у розгортанні та підтримці сервісу в різних середовищах.

Окремо слід враховувати технічні аспекти інтеграції зі сторонніми сервісами, зокрема авторизацією користувача, обробкою інтелектуальних запитів і зберіганням файлів. Також система повинна забезпечувати безперебійну роботу у випадку збоїв на стороні користувача або серверного обладнання, що досягається шляхом впровадження відповідних механізмів відновлення та логування.

2.1.2 Аналіз функціональних вимог сервісу

Функціональні вимоги визначають, які задачі має виконувати система, та які можливості повинні бути доступні користувачам для досягнення цілей взаємодії з вебдодатком. Основною функціональністю є генерація персоналізованої програми тренувань на основі вхідних параметрів користувача, таких як ціль тренування, рівень фізичної підготовки, тривалість занять та інші характеристики.

Однією з ключових вимог є надання зручного інтерфейсу для заповнення первинної анкети, яка дозволяє системі зібрати всю необхідну

інформацію для подальшої генерації індивідуального плану. Після заповнення анкети користувач отримує доступ до генерованої програми з можливістю переглядати вправи, слідкувати за прогресом та редагувати свої цілі. Кожна вправа супроводжується текстовим описом та ілюстративним матеріалом.

Додаток повинен забезпечувати можливість перегляду історії тренувань та фіксації результатів, що дає змогу користувачу аналізувати динаміку змін. Також важливо реалізувати механізм планування, який дозволяє відображати тренування у вигляді календаря з можливістю встановлення нагадувань або редагування розкладу. Користувач має отримувати актуальну інформацію про заплановані або виконані заняття.

Крім базової функціональності, система повинна підтримувати автентифікацію користувачів, персональний кабінет, збереження даних у хмарній базі та динамічне оновлення тренувального плану у випадку зміни введених параметрів. Усі функціональні компоненти повинні працювати узгоджено, забезпечуючи цілісну та ефективну взаємодію з сервісом.

2.1.3 Аналіз вимог до інтерфейсу користувача

Вимоги до інтерфейсу користувача відіграють важливу роль у забезпеченні зручності взаємодії з вебдодатком. Інтерфейс повинен бути інтуїтивно зрозумілим, візуально привабливим та адаптивним до різних розмірів екранів. Особливої уваги потребує чітка навігація між основними розділами сервісу: головна сторінка, персональна програма тренувань, трекер прогресу та налаштування облікового запису.

Сторінка заповнення початкових параметрів повинна містити логічно згруповані поля з чіткими підказками, які полегшують процес введення даних. Користувач має розуміти, як саме його вибір впливає на результат, тому важливо забезпечити зворотний зв'язок ще до моменту генерації

програми. Додатково рекомендується використовувати візуальні елементи, які покращують сприйняття процесу налаштування.

Візуалізація тренувального плану має бути реалізована у вигляді структури, яка поєднує текстову інформацію з графічним представленням вправ. Для кожної вправи повинна відображатись основна інформація: назва, опис, кількість повторень, час виконання та супровідне зображення або ілюстрація. Важливо, щоб усі елементи інтерфейсу не перевантажували користувача інформацією, а сприяли ефективному орієнтуванню.

Адаптивність дизайну забезпечує можливість комфортного використання сервісу як на стаціонарних комп'ютерах, так і на мобільних пристроях. Інтерфейс має залишатися функціональним незалежно від типу екрана, а всі інтерактивні елементи повинні бути легкими у використанні. Це дозволить залучити ширше коло користувачів, включаючи тих, хто віддає перевагу тренуванням з мобільного телефону.

2.2 Розробка функціональної моделі системи

Функціональна модель системи описує логіку роботи сервісу з позиції основних дій користувача та взаємодії між ключовими компонентами. Метою побудови такої моделі є виявлення всіх можливих сценаріїв використання, структуризація функціональних блоків і формування чіткої послідовності процесів, які забезпечують виконання цільових задач. Це дозволяє мінімізувати неузгодженості під час розробки та закласти основу для реалізації узгодженої архітектури.

Одним із центральних елементів функціональної моделі є початковий сценарій, у межах якого користувач реєструється або входить у систему, після чого заповнює коротку анкету з інформацією про ціль тренувань, рівень фізичної підготовки, доступний час, а також інші параметри. На основі цих даних формується персоналізована програма, яку користувач

може переглядати, редагувати та запускати. Передбачена також можливість відстеження прогресу через окремий блок трекінгу.

Для зручності сприйняття функціональна модель розбивається на кілька блоків: модуль автентифікації, модуль обробки користувацьких параметрів, модуль генерації тренувального плану, модуль перегляду історії занять та модуль візуалізації вправ. Кожен з них виконує свою роль у забезпеченні повноцінного функціонування системи. Їх взаємодія дозволяє реалізувати повний цикл роботи – від введення даних до завершення тренування.

У межах даного підрозділу також сформовано загальну схему використання сервісу, яка демонструє основні переходи між етапами роботи. Завдяки побудові функціональної моделі можна чітко визначити, які саме компоненти мають бути реалізовані у наступних етапах розробки, а також які API-запити необхідно забезпечити між клієнтською та серверною частинами. Це підвищує узгодженість та керованість проєкту.

2.2.1 Визначення основних сценаріїв використання

Основні сценарії використання системи визначають послідовність дій, які виконує користувач для досягнення певної мети у межах вебдодатку. Найбільш базовим сценарієм є реєстрація або вхід до особистого кабінету, що відкриває доступ до персонального функціоналу. Після автентифікації користувач може розпочати налаштування свого тренувального плану, заповнивши анкету з початковими параметрами.

Наступний типовий сценарій включає генерацію індивідуального плану тренувань. Користувач вказує свою мету, доступний графік, рівень підготовки та обирає, які типи тренувань його цікавлять. Після цього система формує відповідну програму, яку можна переглядати по днях або категоріях. У кожному тренуванні зазначено вправи, тривалість, кількість підходів та повторень.

Ще один важливий сценарій – це перегляд та фіксація виконаних тренувань. Користувач відзначає, яке тренування було завершено, після чого оновлюється статистика в особистому кабінеті. Це дозволяє аналізувати прогрес і бачити результати у вигляді графіків або списків. Такий підхід сприяє підвищенню мотивації та самодисципліни.

Окремим сценарієм є редагування параметрів користувача. Якщо змінюється мета або фізичні можливості, користувач має змогу оновити анкету, і система сформує новий план. Таким чином, сценарії використання охоплюють повний життєвий цикл взаємодії з додатком – від входу до системи до постійної роботи з тренуваннями та персональними даними.

2.2.2 Формування функціональних блоків системи

Формування функціональних блоків системи дозволяє структуровано представити основні компоненти сервісу та їх взаємодію. У межах даного проекту кожен блок (рисунок 2.1) виконує чітко визначену роль, що забезпечує модульність, простоту в розробці та можливість подальшого розширення. Такий підхід дозволяє розділити логіку клієнтської і серверної частин та визначити зовнішні інтеграції, зокрема з OpenAI API.

Першим функціональним блоком є модуль автентифікації, який реалізується через інтеграцію з сервісом Auth0. Він відповідає за реєстрацію, вхід до системи, зберігання токена та контроль доступу до приватного функціоналу. Наступний блок – це модуль збору параметрів користувача, який дозволяє в інтерактивному форматі ввести особисті дані, обрати мету тренувань, рівень складності, тривалість тощо.

Ключовим блоком системи є генератор персоналізованої програми тренувань, який взаємодіє з OpenAI API. На основі введених параметрів користувача сервер формує запит до мовної моделі, що повертає індивідуально згенеровану програму у текстовому форматі. Цей блок не використовує попередньо заготовлені шаблони або базу вправ, а

покладається на інтелектуальну обробку запиту та контексту за допомогою штучного інтелекту.

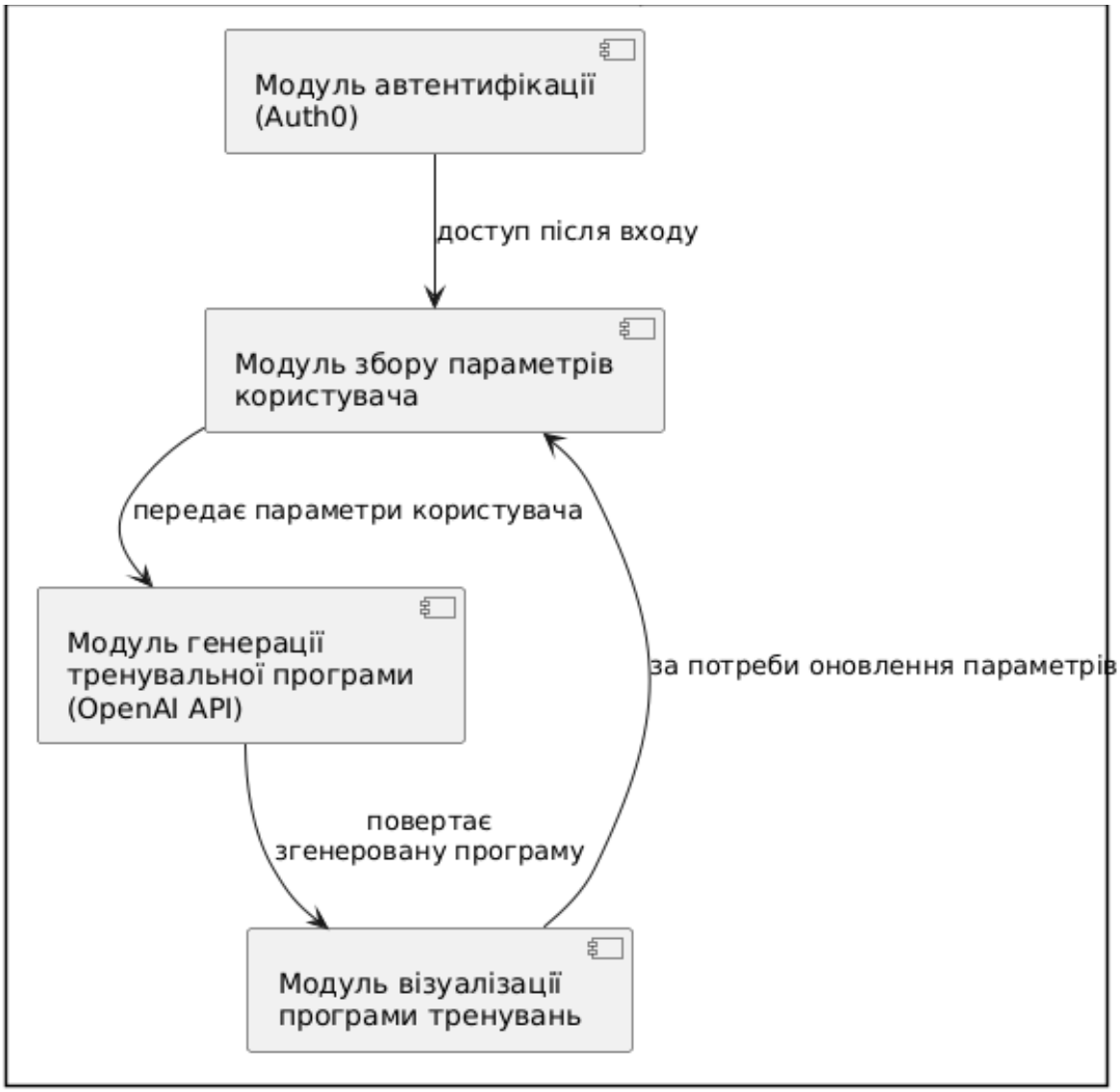

Рисунок 2.1 – Взаємодія функціональних блоків

Окремий блок відповідає за візуалізацію отриманого плану на клієнтській частині. Він дозволяє переглядати програму, структурувати її по днях або типах навантажень, а також фіксувати виконання вправ. Цей модуль тісно пов’язаний із користувацьким інтерфейсом і забезпечує повноцінну взаємодію між людиною та згенерованим вмістом.

2.2.3 Схема використання сервісу

Схема використання сервісу описує базову логіку взаємодії користувача з системою та послідовність обміну даними між її ключовими компонентами. Така схема дає змогу узагальнено представити основні етапи, які проходить користувач, починаючи з моменту відкриття вебдодатку і закінчуючи отриманням персоналізованого тренувального плану.

Далі можна побачити діаграму послідовності, яка ілюструє типовий сценарій взаємодії користувача з сервісом (рисунок 2.2).

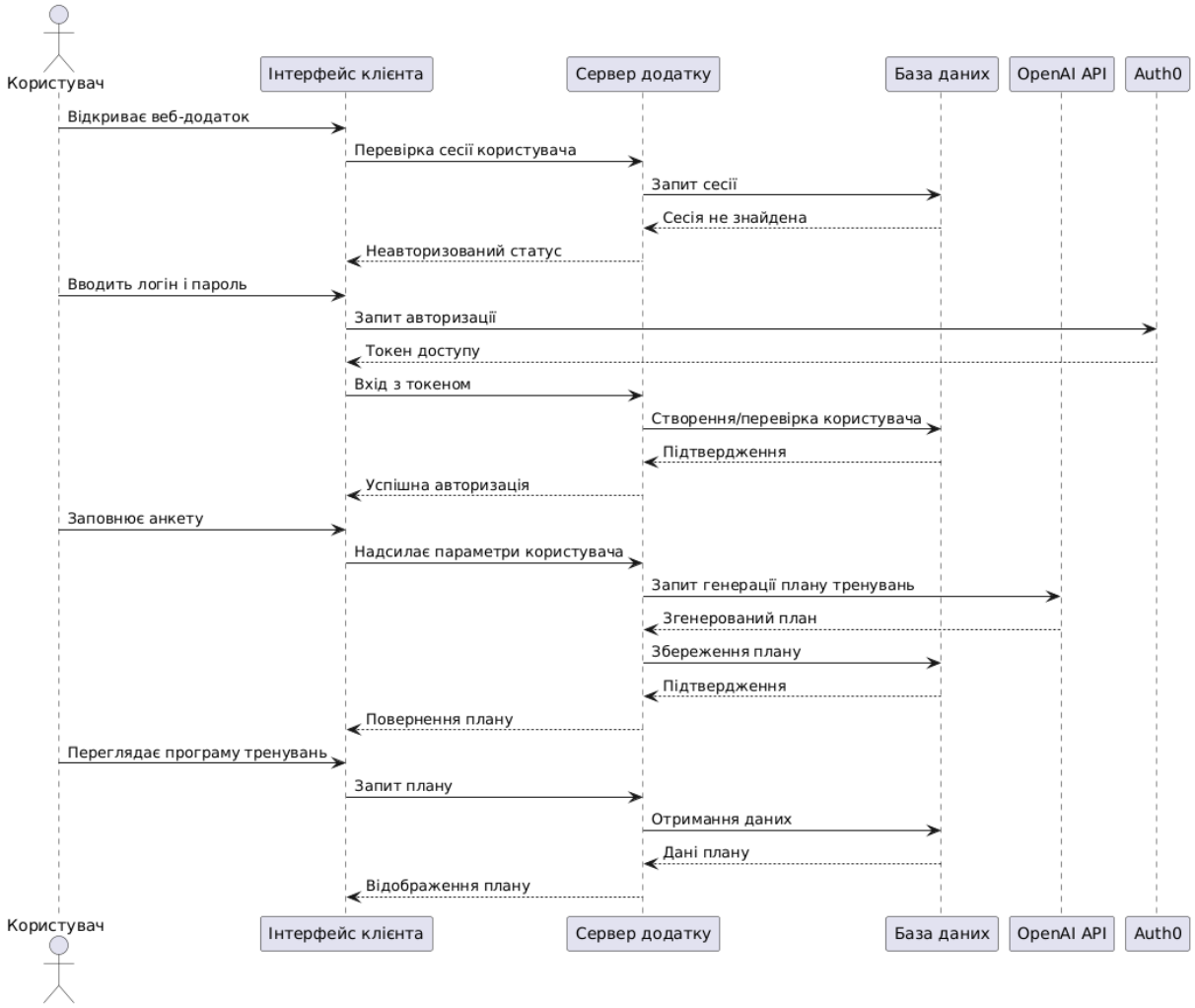

Рисунок 2.2 – Схема використання сервісу

Після відкриття сторінки відбувається перевірка сесії, і в разі її відсутності користувач проходить автентифікацію через Auth0. Далі він заповнює анкету, яка надсилається на сервер. Сервер, у свою чергу, звертається до OpenAI API для генерації індивідуального плану тренувань. Отриманий результат зберігається у базі даних і повертається клієнту для візуалізації.

Ця схема дозволяє зрозуміти взаємозв'язки між клієнтською частиною, сервером, базою даних та сторонніми сервісами, що беруть участь у побудові логіки додатку. Такий підхід до моделювання забезпечує повну картину роботи системи з урахуванням усіх ключових технічних складових, що критично важливо для подальшого проектування і тестування.

2.3 Розробка архітектури

Архітектура системи визначає загальну логіку побудови програмного забезпечення, включаючи взаємодію між її ключовими компонентами. У межах вебдодатку для створення персональних програм тренувань архітектурне проектування дозволяє чітко розмежувати клієнтську та серверну частини, а також зовнішні сервіси, що забезпечують генерацію вмісту або обробку користувацьких даних. Такий підхід сприяє підвищенню масштабованості, стабільності й безпеки системи.

Архітектура сервісу базується на принципах модульності та розподіленості, що дозволяє кожному компоненту виконувати вузькоспеціалізовану задачу. У результаті досягається зменшення залежностей між модулями та спрощення процесу тестування. Клієнтська частина відповідає за взаємодію з користувачем, серверна – за обробку запитів, логіку бізнес-процесів та інтеграцію з OpenAI API, а хмарна інфраструктура виконує роль середовища зберігання і розгортання.

У цьому підрозділі представлено загальне бачення архітектури та описано її основні складові. Деталізація виконуватиметься у наступних пунктах, де буде розглянуто логіку побудови клієнтської частини, структуру серверної взаємодії та особливості підключення до зовнішніх інструментів. Це дасть змогу узгодити всі технічні елементи у єдину функціональну систему.

2.3.1 Клієнтська архітектура

Клієнтська архітектура вебдодатку базується на фреймворку React, який реалізує компонентний підхід до побудови інтерфейсу. Це дозволяє розділити додаток на незалежні частини, кожна з яких відповідає за окрему функціональність. Така модульність полегшує підтримку та тестування, а також дає можливість гнучко масштабувати інтерфейс у майбутньому.

Навігація між сторінками реалізована за допомогою бібліотеки React Router. Вона забезпечує швидке перемикання між основними розділами без перезавантаження сторінки, що покращує користувацький досвід. У межах маршрутизації організовано доступ до таких ключових сторінок, як вхід, анкета користувача, перегляд тренувального плану та статистика.

Для керування формами введення та валідацією даних використовується бібліотека React Hook Form. Це рішення забезпечує ефективну обробку полів форми, дозволяє уникати зайвих ререндерів і зменшує навантаження на інтерфейс. Користувачі мають змогу легко вводити необхідні параметри, а система одразу реагує на помилки чи відсутність обов’язкових полів.

Інтерфейс побудовано на основі бібліотеки Joy UI, яка дозволяє швидко створювати естетичні та адаптивні компоненти. Вона підтримує темізацію, типографіку, колірні схеми та адаптацію до різних розмірів екрана, що критично важливо для підтримки мобільних пристроїв. Кожен

елемент інтерфейсу має продумане візуальне оформлення та інтуїтивно зрозумілу поведінку.

Стан додатку зберігається у локальному контексті та при потребі – у хмарі, якщо користувач авторизований. Це дозволяє не втрачати дані між сеансами, а також синхронізувати історію взаємодії з сервером. Для передачі даних на сервер використовуються асинхронні запити, що забезпечує безперебійну роботу інтерфейсу навіть у разі затримок з боку бекенду.

Загалом клієнтська архітектура реалізована таким чином, щоб забезпечити високу швидкодію, гнучкість і масштабованість. Кожен компонент працює автономно, але в тісному зв’язку з іншими, що формує цілісну систему взаємодії користувача з сервісом. Такий підхід відповідає сучасним стандартам веброзробки та гарантує зручність у використанні.

2.3.2 Серверна архітектура

Серверна архітектура системи побудована на основі фреймворку Nest.js, який використовує мову TypeScript та дозволяє реалізовувати модульну й масштабовану структуру вебдодатку. Nest.js базується на принципах архітектури REST API та надає зручні механізми для організації контролерів, сервісів і репозиторіїв. Завдяки цьому серверна частина системи є розширюваною, надійною та легкою для підтримки.

Основні компоненти серверної частини поділені на окремі модулі, які виконують чітко визначені завдання. Контролери відповідають за прийом HTTP-запитів від клієнта, виклик відповідних сервісів та формування відповіді. Сервіси реалізують бізнес-логіку обробки параметрів користувача, генерації запитів до OpenAI API, а також збереження отриманих даних у відповідних структурах.

Інтеграція з OpenAI API є ключовою частиною серверної архітектури. В межах одного з сервісів реалізовано логіку формування динамічного

запиту до мовної моделі на основі введених користувачем параметрів. Отримана відповідь проходить обробку і перетворюється у зручний формат для виводу на клієнтському інтерфейсі. Це забезпечує унікальність і персоналізацію кожного тренувального плану.

Сервер також здійснює авторизацію користувачів через інтеграцію з Auth0. Після проходження аутентифікації клієнт надсилає токен доступу, який перевіряється сервером. Успішна перевірка надає користувачу доступ до захищених ресурсів, включно з генерацією програм та збереженням прогресу. Цей підхід гарантує безпеку персональних даних.

Хоча система не використовує класичну базу даних для зберігання вправ, серверна частина все одно відповідає за управління сеансами, параметрами запитів, журналами помилок та іншою технічною інформацією. Це дозволяє покращити контроль за виконанням запитів, виявляти помилки та проводити аудит взаємодії з OpenAI API.

Архітектура сервера побудована з урахуванням можливості масштабування. Завдяки підтримці контейнеризації за допомогою Docker та розгортанню на Google Cloud, система може обробляти зростаючу кількість запитів, адаптуючись до навантаження. Такий підхід гарантує стабільність і готовність до майбутнього розширення функціоналу.

2.4 Розробка бази даних

Проектування структури даних є важливим етапом розробки системи, особливо коли йдеться про динамічний і персоналізований контент. У межах цього проекту було прийнято рішення використовувати нереляційну базу даних Google Firestore, яка забезпечує гнучке зберігання документів у вигляді колекцій. Такий підхід дозволяє легко масштабувати систему, уникати фіксованих схем таблиць і оперативно оновлювати структуру даних відповідно до змін у логіці програми.

Firestore організовано навколо кількох основних колекцій: користувачів, анкетних даних, згенерованих тренувальних програм і журналів активності. Кожен документ містить унікальний ідентифікатор та набір ключових атрибутів. Далі можна побачити логічну структуру зв’язків між основними колекціями, яка ілюструє, як взаємодіють дані на рівні зберігання (рисунок 2.3). Наприклад, анкета користувача пов’язана з його ідентифікатором, а згенерований план тренувань базується на цій анкеті. Журнали активності фіксують взаємодію користувача з кожним планом.

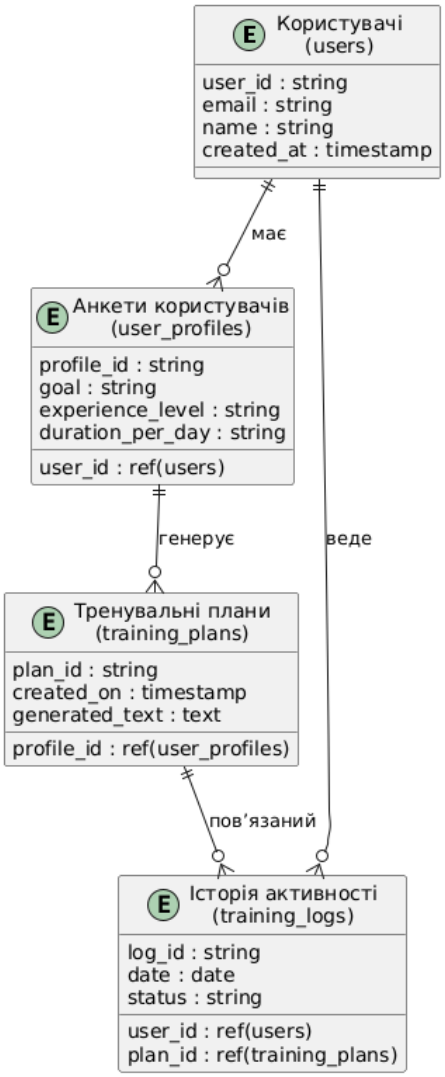

Рисунок 2.3 – Логічна структура зберігання даних у Firestore

Оскільки вправи формуються динамічно через OpenAI API, у базі немає попередньо створених шаблонів або структур вправ. Замість цього зберігається вже згенерований текстовий результат разом з контекстною інформацією, яка використовувалася під час генерації. Це дозволяє при повторному запиті аналізувати історію взаємодії, змінювати параметри та отримувати нові варіанти тренувань.

Обрана модель зберігання дозволяє легко реалізувати фільтрацію, історію тренувань, перегляд прогресу, а також здійснювати персоналізовані оновлення плану. Завдяки гнучкості Firestore система може адаптуватися до нових функціональних вимог без суттєвого рефакторингу існуючої структури. Такий підхід забезпечує ефективність, масштабованість і швидкість доступу до даних.

Окрім основної структури, реалізація бази даних у Firestore дозволяє використовувати вбудовані механізми безпеки, такі як правила доступу на рівні документів. Це дає змогу обмежити доступ до певних даних лише для авторизованих користувачів, а також забезпечити, щоб кожен користувач мав змогу переглядати лише власні анкети, тренувальні плани та історію активності. Такий рівень ізоляції й контролю відповідає сучасним вимогам до захисту персональних даних і підвищує довіру до сервісу.

2.5 Вибір інструментів реалізації

Під час реалізації системи було обрано стек технологій, який забезпечує високу продуктивність, гнучкість у розробці та легкість у масштабуванні. Основу клієнтської частини складає фреймворк React, що дозволяє створювати адаптивні інтерфейси з компонентною структурою. Цей підхід забезпечує зручність повторного використання логіки та швидке рендерення елементів сторінки.

Для стилізації інтерфейсу використовується бібліотека Joy UI, яка поєднує в собі мінімалістичний дизайн та широкі можливості налаштування

зовнішнього вигляду компонентів. Це дозволяє створювати зручний і привабливий інтерфейс без необхідності розробляти стилі з нуля. Взаємодія між компонентами керується за допомогою бібліотеки React Hook Form, яка спрощує обробку форм, валідацію введених даних та контроль станів.

Для організації навігації використовується React Router, який реалізує маршрутизацію на клієнтському рівні та дозволяє реалізувати багатосторінкову логіку без перезавантаження сторінки. Для безпечної автентифікації інтегровано сервіс Auth0, що дозволяє організувати реєстрацію, вхід та управління сесіями без потреби створювати власну систему зберігання паролів.

Серверна частина побудована на основі фреймворку Nest.js, що реалізований на мові програмування TypeScript. Його модульна архітектура дозволяє ефективно структурувати бізнес-логіку, маршрути API, сервіси та репозиторії. Nest.js підтримує використання декораторів, ін’єкцію залежностей та асинхронне програмування, що робить його придатним для побудови масштабованих вебсервісів.

Дані користувачів, а також інформація про тренування, зберігаються у хмарній базі даних Google Firestore, яка є частиною екосистеми Firebase. Цей сервіс дозволяє організувати зберігання у форматі документів, забезпечуючи високу швидкість доступу до даних та їх синхронізацію в реальному часі. Для роботи з Firebase застосовується офіційний SDK, який полегшує взаємодію з базою, автентифікацією та сховищем.

Фінальне розгортання сервісу здійснюється на платформі Google Cloud з використанням Docker, що дозволяє ізолювати середовище розробки та запуску, забезпечити повторюваність результатів і зменшити ризики помилок при міграції між середовищами. Використання контейнеризації гарантує, що додаток працюватиме стабільно незалежно від хостингу чи інфраструктури.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Загальний опис застосованих технологій

У процесі реалізації вебдодатку для створення персоналізованих тренувальних програм було використано сучасний набір технологій, що охоплює як клієнтську, так і серверну частину системи. Такий стек інструментів обрано з урахуванням вимог до продуктивності, адаптивності, масштабованості, безпеки та швидкої розробки. Особливу увагу приділено інтеграції зі сторонніми сервісами, зокрема OpenAI API, Auth0, Google Cloud Firestore, а також підтримці контейнеризації для хмарного розгортання. Використання перевірених інструментів дозволило забезпечити високу якість, стабільність та гнучкість архітектури системи.

Клієнтська частина реалізована з використанням сучасної бібліотеки React – популярного JavaScript-фреймворку, що базується на компонентному підході. Така структура дозволяє розділяти функціональність інтерфейсу на окремі незалежні модулі, які легко тестуються, масштабуються та повторно використовуються. React забезпечує швидку реакцію на дії користувача, мінімізує оновлення сторінки та підвищує загальну продуктивність додатку. Цей фреймворк активно підтримується спільнотою, має великий вибір готових рішень і добре документований.

Для стилізації інтерфейсу використано бібліотеку Joy UI, яка є частиною екосистеми MUI. Вона дозволяє реалізувати адаптивний та естетично привабливий дизайн з використанням готових компонентів, що мають широкі можливості для кастомізації. Joy UI підтримує типографіку, гнучке управління кольоровими схемами та добре поєднується з компонентною логікою React. Завдяки цьому створення користувацького інтерфейсу відбувалося швидко та узгоджено з принципами сучасного вебдизайну.

Для маршрутизації у межах клієнтської частини використано бібліотеку React Router. Вона забезпечує переходи між сторінками без повного перезавантаження, зберігаючи при цьому поточний стан додатку. Це дозволяє реалізувати багатосторінкову логіку роботи, включаючи окремі маршрути для форми реєстрації, заповнення анкети, перегляду згенерованої програми тренувань, а також перегляду прогресу. Такий підхід відповідає потребам інтерактивних додатків, які мають складну навігацію.

Для керування формами, обробки даних користувача та валідації було використано бібліотеку React Hook Form. Вона забезпечує мінімальне споживання ресурсів, високу продуктивність та можливість централізованого контролю стану форми. Користувачі мають змогу вводити свої параметри через зручні поля, а система миттєво реагує на введення некоректних або неповних даних. Це підвищує точність введеної інформації та покращує досвід взаємодії з інтерфейсом.

Для реалізації авторизації використано платформу Auth0 – надійний сервіс управління ідентифікацією та доступом. Auth0 забезпечує просту інтеграцію механізмів реєстрації, входу та управління сесіями без потреби реалізовувати власну систему зберігання облікових записів. Увесь процес відбувається за допомогою токенів, що відповідає сучасним стандартам безпеки і дозволяє захищати персональні дані користувачів навіть у масштабованих хмарних середовищах.

Серверна частина системи побудована з використанням фреймворку Nest.js, що базується на TypeScript і підтримує архітектурні підходи, характерні для об’єктно-орієнтованого програмування. Nest.js забезпечує модульність, зручну структуру сервісів і контролерів, підтримку декораторів та ін’єкцію залежностей. Це дозволило реалізувати чітке розділення логіки генерації запитів до OpenAI API, обробки параметрів користувача та взаємодії з хмарною базою Firestore. Завдяки Nest.js вдалося досягти узгодженості коду, зменшити дублювання та спростити обробку запитів.

Для зберігання даних використано хмарну нереляційну базу Google Firestore, яка є частиною платформи Firebase. Firestore підтримує модель документоорієнтованого зберігання, що дозволяє динамічно зберігати змінні об’єкти без фіксованої структури. У межах проєкту Firestore використовується для збереження інформації про користувачів, параметри, введені ними під час заповнення анкети, та результати згенерованих тренувальних програм. Перевагою є можливість синхронізації в реальному часі та легкість у масштабуванні системи без необхідності розгортати власні сервери баз даних.

Генерація індивідуальних тренувальних програм реалізована через інтеграцію з OpenAI API. Цей інструмент дозволяє обробляти текстові запити, сформовані на основі параметрів, введених користувачем, і генерувати змістовні тренувальні описи у відповідному стилі. Таке рішення дозволяє уникнути зберігання шаблонів або бази вправ, а натомість формувати унікальний результат при кожному запиті, що значно підвищує рівень персоналізації.

Усі серверні компоненти розгорнуто за допомогою Docker, що дозволяє створювати ізольоване середовище виконання і гарантувати однакову роботу програми у різних умовах. Контейнеризація спрощує деплой, оновлення та масштабування сервісу. Додаток було розгорнуто на платформі Google Cloud, що забезпечує надійність, відмовостійкість і високу швидкість обробки запитів навіть при збільшенні навантаження.

Завдяки обраному стеку технологій розроблений вебдодаток поєднує у собі продуктивність, зручність, гнучкість та потенціал до подальшого розширення. Кожна складова технологічної інфраструктури сприяє досягненню цілей, поставлених у межах дипломного проєкту, і забезпечує стабільну роботу інтелектуального фітнес-сервісу в умовах реального використання.

3.2 Реалізація сервісу

Реалізація створення користувача починається з отримання аватару за URL та збереження його у хмарне сховище Firebase Storage у вигляді зображення, прив’язаного до електронної адреси користувача. Далі формується пряме посилання на це зображення, яке буде використовуватись як значення поля avatar. У межах транзакції перевіряється, чи існує користувач з таким Auth0 ID, і якщо ні – створюється новий документ у колекції users з базовими атрибутами, включно з URL аватару. Увесь процес подано у лістингу 3.1.

Лістинг 3.1 – Програмний код, що реалізує створення нового користувача після авторизації через Auth0

```
@Injectable()
export class AuthService {
 constructor(private readonly firebase: FirebaseService) {}

  async createAuth0User({ auth0Id, email, name, avatar }: CreateAuth0User): Promise<void> {
    const fileName = `avatars/${Buffer.from(email).toString('base64')}.jpg`;
    const fileUrl = `https://storage.googleapis.com/bucket/${fileName}`;
    const avatarData = await fetch(avatar).then((r) => r.arrayBuffer());
    await getStorage().bucket().file(fileName).save(Buffer.from(avatarData));

    await firestore().runTransaction(async (tx) => {
      const existing = await tx.get(
        firestore().collection('users').where('auth0Id', '==', auth0Id));
```

Продовження лістингу 3.1

```
          if (!existing.empty) throw new Error('User already
exists');

     tx.create(this.firebase.getFirestore().collection('users')
.doc(), {
              auth0Id,
              email,
              name,
              avatar: fileUrl,
              isOnboarded: false,
            });
         });
       }
     }
```

У цьому коді відбувається ініціалізація Firebase на основі облікових даних, отриманих із конфігураційного сервісу, що дозволяє безпечно зберігати ключі доступу в середовищі. Після ініціалізації додаток отримує доступ до бази даних Firestore та хмарного сховища Google Cloud Storage через відповідні сервіси. Створений сервіс надає методи для зчитування екземплярів Firestore і сховища, як показано в лістингу 3.2.

Лістинг 3.2 – Програмний код, що ініціалізує підключення до Firebase за допомогою конфігурації з середовища та надає доступ до Firestore і хмарного сховища Google Cloud Storage

```
     @Injectable()
     export class FirebaseService {
       private firestore: admin.firestore.Firestore;
       private       bucket:       ReturnType<ReturnType<typeof
getStorage>['bucket']>;

       constructor(config: ConfigService) {
```

Продовження лістингу 3.2

```
        const credentials = {
          projectId: config.get('FIREBASE_PROJECT_ID'),
          privateKey:
config.get('FIREBASE_PRIVATE_KEY')?.replace(/\\n/g, '\n'),
          clientEmail: config.get('FIREBASE_CLIENT_EMAIL'),
        };

        admin.initializeApp({
          credential: admin.credential.cert(credentials),
          databaseURL:
`https://${credentials.projectId}.firebaseio.com`,
          storageBucket:
`${credentials.projectId}.firebasestorage.app`,});

        this.firestore = admin.firestore();
        this.bucket = getStorage().bucket();
      }

      getFirestore() {
        return this.firestore;
      }

      getBucket() {
        return this.bucket;
      }
    }
```

У цьому коді створюється екземпляр клієнта OpenAI, який використовується для надсилання запитів до відповідного API. Ініціалізація здійснюється за допомогою ключа доступу, що зчитується з конфігурації середовища. Готовий об’єкт клієнта доступний через метод getOpenAI, як подано у лістингу 3.3.

Лістинг 3.3 – Програмний код, що ініціалізує клієнт OpenAI для взаємодії з API на основі ключа з конфігурації середовища

```
@Injectable()
export class OpenAIService {
  private openAI: OpenAI;

  constructor(config: ConfigService) {
    this.openAI = new OpenAI({
      apiKey: config.get('OPENAI_API_KEY'),
    });
  }

  getOpenAI() {
    return this.openAI;
  }
}
```

У цьому фрагменті реалізовано контролер, який відповідає за обробку запитів, пов’язаних з інформацією про користувача, а також за ініціалізацію його взаємодії із системою. Як видно з лістингу 3.4, методи контролера захищені за допомогою механізму авторизації JWT, що гарантує, що до них можуть звертатися лише автентифіковані користувачі. Кожен маршрут відповідає за конкретну дію – отримання поточного користувача, онбординг, оновлення ваги, відзначення виконаного тренування або застосування згенерованого плану вправ.

Контролер отримує необхідні дані через тіла HTTP-запитів або параметри маршрутів та делегує бізнес-логіку відповідним методам сервісу UserService. Завдяки цьому дотримано принципу розділення відповідальностей – контролер лише обробляє запити та формує відповідь, а логіка роботи з базою даних зосереджена на рівні сервісу. Такий підхід дозволяє спростити тестування і підтримку системи.

Лістинг 3.4 – Програмний код, що реалізує контролер для взаємодії з користувачем

```
@Controller('users')
export class UserController {
  constructor(private readonly userService: UserService) {}

  @UseGuards(AuthGuard('jwt'))
  @Get('/current')
  async getCurrentUser(@CurrentUser() { id }: CurrentUserType) {
    return await this.userService.getUserByAuth0Id(id);
  }

  @UseGuards(AuthGuard('jwt'))
  @Post('/current/onboard')
  async onboardUser(@CurrentUser() { id }: CurrentUserType, @Body() data: OnboardUserInput) {
    return await this.userService.onboardUser(id, data);
  }

  @UseGuards(AuthGuard('jwt'))
  @Post('/current/weight')
  async updateWeight(@Body() input: UpdateWeightInput, @CurrentUser() { id }: CurrentUserType) {
    return await this.userService.updateWeight({ auth0Id: id, ...input });
  }

  @UseGuards(AuthGuard('jwt'))
  @Post('/current/drills/:id')
  async markDrillFinished(@CurrentUser() { id }: CurrentUserType, @Param('id') drillId: string) {
    return await this.userService.markDrillFinished({ auth0Id: id, drillId });
  }
```

Продовження лістингу 3.4

```
  @UseGuards(AuthGuard('jwt'))
  @Post('/current/drills')
  async applyDrillsPlan(@CurrentUser() { id }:
CurrentUserType, @Body() data: ApplyDrillsPlanInput)
   {
    return await this.userService.applyDrillsPlan
   ({
auth0Id: id, ...data
   });
  }
}
```

На цьому етапі реалізується основна логіка роботи з користувачами, включно з отриманням, онбордингом, оновленням фізичних показників і застосуванням плану тренувань.

Як видно з лістингу 3.5, усі ці дії виконуються через транзакції Firestore, що гарантує цілісність і узгодженість змін, навіть у випадку паралельного доступу до одних і тих самих записів. Особливу увагу приділено перевірці існування користувача, запобіганню повторному онбордингу, а також збереженню історичних даних про вагу та відсоток жиру.

Для кожної з функцій використовується окрема приватна логіка, зокрема метод _getUserDoc спрощує доступ до документа користувача в рамках транзакції, а _generateDrills відповідає за створення нового масиву вправ з урахуванням налаштувань.

Завдяки чіткому поділу відповідальностей та структурованості коду сервіс забезпечує стабільне функціонування користувацької частини додатку, а також створює основу для подальшого розширення логіки без порушення існуючих механізмів.

Лістинг 3.5 – Програмний код, що реалізує логіку роботи з користувачем

```
@Injectable()
export class UserService {
  constructor(
    private readonly firebase: FirebaseService,
    private readonly config: RemoteConfigService,
  ) {}

  async getUserByAuth0Id(auth0Id: string): Promise<User | null> {
    const snapshot = await this.firebase.getFirestore()
      .collection('users')
      .where('auth0Id', '==', auth0Id)
      .limit(1)
      .get();

    if (snapshot.empty) return null;
    return mapUserDtoToUser({ id: snapshot.docs[0].id, ...snapshot.docs[0].data() });
  }

  async onboardUser(auth0Id: string, values: OnboardValues): Promise<User | null> {
    const user = await this.getUserByAuth0Id(auth0Id);
    if (!user || user.isOnboarded) throw new Error('User not found or already onboarded');

    const drills = await this.config.getDefaultDrillsConfig();
    await this.firebase.getFirestore().collection('users').doc(user.id).update({
      ...values,
      isOnboarded: true,
```

Продовження лістингу 3.5

```
            drills: drills.map(...), // map to dto
          });

          return await this.getUserByAuth0Id(auth0Id);
        }

        async applyDrillsPlan({ auth0Id, plan, durationInWeeks,
updateCurrentWeek }): Promise<User> {
          await
this.firebase.getFirestore().runTransaction(async (tx) => {
            const userDoc = await this._getUserDoc(tx, auth0Id);
            const    newDrills    =    this._generateDrills(plan,
durationInWeeks, updateCurrentWeek);
            tx.update(userDoc.ref, { drills: newDrills.map(...)
});
          });

          return await this.getUserByAuth0Id(auth0Id);
        }

        async    markDrillFinished({    auth0Id,    drillId    }):
Promise<User> {
          await
this.firebase.getFirestore().runTransaction(async (tx) => {
            const userDoc = await this._getUserDoc(tx, auth0Id);
            const drills = [...]; // mark drill as finished
            tx.update(userDoc.ref, { drills });
          });

          return await this.getUserByAuth0Id(auth0Id);
        }
        }
      }
```

Контролер, представлений у лістингу 3.6, відповідає за обробку запиту користувача на створення індивідуального плану тренувань. Він перевіряє, чи існує користувач у базі даних та чи пройшов він процес первинного налаштування. Якщо умови виконані, викликається відповідний сервіс для генерації плану. Усі необхідні параметри для генерації передаються на основі збереженої інформації про користувача.

Лістинг 3.6 – Програмний код, що обробляє запит авторизованого користувача на генерацію плану тренувань

```
@Controller('drills')
export class DrillsController {
  constructor(
    private readonly userService: UserService,
    private readonly drillsService: DrillsService,) {}
  @Post('/generate')
  async generateDrillsPlan(
    @CurrentUser() { id }: CurrentUserType,
    @Body() input: GenerateDrillsPlanInput,
  ): Promise<Drill[]> {
    const user = await this.userService.getUserByAuth0Id(id);}
    return this.drillsService.generateDrillsPlan({
      ...input,
      duration: 'month',
      experience: user.experience,
      goal: user.goal,
      restrictionsOrInjuries: user.restrictionsOrInjuries,
      age: user.age,
      height: user.height,
      weight: user.weight,
      fatPercentage: user.fatPercentage,
      sex: user.sex,
    });
```

Реалізація логіки генерації тренувального плану за допомогою OpenAI API представлена у лістингу 3.7. Сервіс отримує персональні параметри користувача, включно зі статтю, віком, ростом, вагою, рівнем підготовки та цільовими м’язовими групами, після чого формує відповідний запит до моделі GPT. Основу повідомлення складає системний інструктаж і користувацький промпт, сформований через функцію buildPrompt, який адаптований до контексту фізичної підготовки.

Важливою особливістю реалізації є підтримка функціональних викликів моделі OpenAI, які дозволяють динамічно збагачувати відповідь медіаданими (відео чи зображеннями) через виклики get_exercise_videos та get_drill_image. В залежності від типу запиту система генерує і відправляє уточнюючі повідомлення, на основі яких відбувається оновлення результатів у два кроки. Завершальна відповідь парситься у форматі JSON, який надалі використовується як структура тренувального плану.

Лістинг 3.7 – Програмний код, що реалізує генерацію персоналізованого плану тренувань через OpenAI API

```
@Injectable()
export class DrillsService {
  constructor(
    private readonly openAI: OpenAIService,
    private readonly config: RemoteConfigService,
  ) {}
  async generateDrillsPlan(input): Promise<Drill[]> {
    const exercises = await this.config.getExercisesConfig();
    const drills = await this.config.getDrillsConfig();
    const messages = [
      { role: 'system', content: 'You are a fitness trainer.' },
      { role: 'user', content: this.buildPrompt(input) },
    ];
```

Продовження лістингу 3.7

```
        const res = await
this.openAI.getOpenAI().chat.completions.create({
          model: 'gpt-4o',
          messages,
          functions,
        });
        return res.choices[0].message;
      }

      private extractJson(text: string): string {
        return text.includes('```json') ? text.slice(7, -3) :
text;
      }
    }
```

На етапі первинного анкетування користувач вводить базові параметри, які проходять перевірку на відповідність визначеним обмеженням. У лістингу 3.8 показано валідацію числових значень для зросту, ваги, віку та жирової маси, включаючи мінімальні та максимальні допустимі межі. Також передбачено контроль правильності введення категоріальних значень, таких як стать, мета тренувань та рівень підготовки. Усі правила валідації супроводжуються повідомленнями, які повертаються користувачеві у разі помилки.

Лістинг 3.8 – Програмний код, що відповідає за валідацію даних, введених користувачем на етапі первинного анкетування у вебдодатку

```
export class OnboardUserInput {
  @IsNumber({}, { message: 'Height must be a number' })
  @Min(50, { message: 'Height must be at least 50 cm' })
  @Max(250, { message: 'Height must be less than 250 cm' })
  height!: number;
  @IsNumber({}, { message: 'Weight must be a number' })
```

Продовження лістингу 3.8

```
    @Min(20, { message: 'Weight must be at least 20 kg' })
    @Max(300, { message: 'Weight must be less than 300 kg' })
    weight!: number;
    @IsNumber({}, { message: 'Age must be a number' })
    @Min(1, { message: 'Age must be at least 1' })
    @Max(120, { message: 'Age must be less than 120' })
    age!: number;
    @IsEnum(['male', 'female'], {
      message: 'Sex must be male, female, or other',
    })
    sex!: Sex;
    @IsEnum(['loose_weight', 'build_muscle', 'stay_fit'], {
      message: 'Goal must be loose_weight, gain_muscle, or
maintain_health',
    })
    goal!: 'loose_weight' | 'build_muscle' | 'stay_fit';

    @IsEnum(['beginner', 'intermediate', 'advanced'], {
      message: 'Experience must be beginner, intermediate, or
advanced',
    })
    experience!: 'beginner' | 'intermediate' | 'advanced';

    @IsOptional()
    @IsString({ message: 'Restrictions or injuries must be a
string' })
    restrictionsOrInjuries: string | null;

    @IsNumber({}, { message: 'Fat percentage must be a number'
    @Min(0, { message: 'Fat percentage must be at least 0' })
    @Max(1, { message: 'Fat percentage must be less than 1'
    fatPercentage!: number;
  }
```

У процесі застосування індивідуального тренувального плану важливо забезпечити коректність структури введених даних. У лістингу 3.9 наведено приклад опису полів, які визначають окрему вправу та тренувальний день, включаючи назву, опис, інструкції, кількість повторів, підходів і час відпочинку. Для кожного поля передбачено обмеження, що контролюють тип даних, граничні значення та максимальну довжину рядків. Також вказано обов'язкові та необов'язкові параметри, такі як зображення вправи або посилання на відеоінструкцію.

Лістинг 3.9 – Програмний код, що описує структуру та валідаційні правила для застосування тренувального плану користувачем

```
export class ExerciseInput {
  @IsString()
  @MaxLength(100, { message: 'Exercise name must be at most
100 characters.' })
  name: string;

  @IsString()
  @MaxLength(500, { message: 'Description must be at most
500 characters.' })
  description: string;

  @IsString()
  @MaxLength(1000, { message: 'Instructions must be at most
1000 characters.' })
  instructions: string;

  @IsInt({ message: 'Sets must be an integer.' })
  @Min(1, { message: 'Sets must be at least 1.' })
  sets: number;

  @IsInt({ message: 'Reps must be an integer.' })
  @Min(1, { message: 'Reps must be at least 1.' })
  reps: number;
```

Продовження лістингу 3.9

```
    @IsInt({ message: 'Rest must be an integer.' })
    @Min(0, { message: 'Rest must be at least 0 seconds.' })
    rest: number;

    @IsOptional()
    @IsUrl({}, { message: 'Link must be a valid URL.' })
    link: string;
  }

  export class DrillInput {
    @IsEnum(Day, {
      message:
        'Day must be one of: monday, tuesday, wednesday,
thursday, friday.',
    })
    day: Day;

    @IsOptional()
    @IsUrl({}, { message: 'Image URL must be a valid URL.' })
    @IsString({ message: 'Image URL must be a string.' })
    imageUrl: string | null;

    @IsString()
    @MaxLength(100, { message: 'Label must be at most 100
characters.' })
    label: string;

    @IsArray({ message: 'Exercises must be an array.' })
    @ValidateNested({ each: true })
    @Type(() => ExerciseInput)
    exercises: ExerciseInput[];
  }
```

У лістингу 3.10 показано налаштування клієнтського API за допомогою бібліотеки Redux Toolkit Query, що забезпечує ефективну взаємодію з бекендом. Конфігурація включає автоматичну передачу токена авторизації в заголовках запитів, а також визначення низки ендпоінтів, необхідних для отримання та зміни інформації про користувача. Зокрема, реалізовано методи для отримання поточного користувача, онбордингу, оновлення параметрів тіла, позначення виконаних вправ, генерації нового плану тренувань та його збереження. Усі запити автоматично оновлюють відповідні дані в кеші завдяки використанню тегів.

Лістинг 3.10 – Програмний код, що налаштовує клієнтське API для вебдодатку на базі Redux Toolkit Query

```
export const api = createApi({
  reducerPath: "mainApi",
  tagTypes: ["CurrentUser"],
  baseQuery: fetchBaseQuery({
    baseUrl: env.VITE_API_URL,
    prepareHeaders: (headers, { getState }) => {
      const token = (getState() as
RootState).auth.accessToken;
      if (token) headers.set("authorization", `Bearer
${token}`);
      return headers;
    },
  }),
  endpoints: (builder) => ({
    getCurrentUser: builder.query({ query: () =>
"users/current", providesTags: ["CurrentUser"] }),
    onboardUser: builder.mutation({
      query: (body) => ({
        url: "users/current/onboard",
        method: "POST",
body: { ...body, fatPercentage: body.fatPercentage / 100}),
```

Продовження лістингу 3.10

```
        invalidatesTags: ["CurrentUser"],
      }),
      updateWeight: builder.mutation({
        query: ({ weight, fatPercentage }) => ({
          url: "users/current/weight",
          method: "POST",
          body: { weight, fatPercentage: fatPercentage/100},
        }),
        invalidatesTags: ["CurrentUser"],
      }),
      markAsFinished: builder.mutation({
        query: (id) => ({ url: `users/current/drills/${id}`,
method: "POST" }),
        invalidatesTags: ["CurrentUser"],
      }),
      generateDrills: builder.mutation({
        query: (body) => ({ url: "/drills/generate", method:
"POST", body }),
        invalidatesTags: ["CurrentUser"],
      }),
      applyDrillsPlan: builder.mutation({
        query: (body) => ({ url: "users/current/drills",
method: "POST", body }),
        invalidatesTags: ["CurrentUser"],
      }),
    }),
  });
```

У лістингу 3.11 представлено реалізацію клієнтської сторінки анкетування користувача, яка відкривається після першої авторизації. Інтерфейс розроблено з використанням компонентів бібліотеки Joy UI та React Hook Form, що забезпечує кероване введення даних і миттєву валідацію введених значень. Користувач заповнює форму, яка включає поля

зросту, ваги, віку, жирового індексу, статі, мети тренувань, рівня фізичної підготовки та можливих обмежень. Після підтвердження форми дані надсилаються через API-запит, а у разі успішного завершення онбордингу користувача перенаправляють на головну сторінку сервісу.

Код також реалізує базову перевірку на помилки, відображає повідомлення про невдалу спробу збереження даних та містить адаптивну візуалізацію компонента. Особливу увагу приділено UX – структура сторінки логічно поділена на блоки заголовка, основної частини з формою та футера, що підвищує зручність користування. Також реалізовано зміну теми інтерфейсу через компонент ColorSchemeToggle. Візуальний стиль інтерфейсу доповнюється фоновим зображенням, яке займає другу половину екрана на десктопних пристроях.

Лістинг 3.11 – Програмний код, що реалізує клієнтський інтерфейс сторінки онбордингу користувача у вебдодатку

```
export const OnboardingPage: FC = () => {
  const [onboardUser, { isLoading, isError }] = useOnboardUserMutation();
  const navigate = useNavigate();
  const {
    control,
    handleSubmit,
    formState: { errors },
  } = useForm<FormState>({
    resolver: zodResolver(schema),
    defaultValues: {
      height: "", weight: "", age: "", fatPercentage: "",
      sex: null, goal: null, experience: null, restrictionsOrInjuries: "",
    },
  });
  return (
    <>
```

Продовження лістингу 3.11

```
          <Box sx={{ width: { xs: "100%", md: "50vw" }, display:
"flex" }}>
            <Box sx={{ minHeight: "100dvh", px: 2 }}>
              <header>
                <Typography                level="title-lg">AI
Coach</Typography>
                <ColorSchemeToggle />
              </header>
              <main>
                <Typography level="h3">A few questions to get
you started</Typography>
                {isError && <Alert type="error" title="Error"
message="Please try again" />}
                <form onSubmit={handleSubmit(onSubmit)}>
                  <InputFormField name="height" label="Height"
control={control} errors={errors} />
                  <InputFormField    name="age"    label="Age"
control={control} errors={errors} />
                  <InputFormField name="weight" label="Weight"
control={control} errors={errors} />
                  <InputFormField         name="fatPercentage"
label="Fat %" control={control} errors={errors} />
                  <RadioGroupFormField  name="sex"  label="Sex"
control={control} errors={errors}>
                    <Radio value="male" label="Male" />
                    <Radio value="female" label="Female" />
                  </RadioGroupFormField>
                  <SelectFormField  name="goal"  label="Goal"
control={control} errors={errors}>
                    <Option          value="loose_weight">Lose
Weight</Option>
                    <Option          value="build_muscle">Gain
Muscle</Option>
```

У лістингу 3.12 реалізовано головну сторінку користувача після проходження онбордингу. Вона відображає загальний стан користувача на поточному тижні, включаючи графіки змін ваги та жирового індексу, а також список запланованих тренувань. Якщо у користувача немає активних тренувань на тиждень, відображається інформативне повідомлення. Також передбачена можливість оновлення ваги й відсотка жиру через модальне вікно, що активується натисканням на відповідні елементи інтерфейсу.

Лістинг 3.12 – Програмний код, що реалізує головну сторінку користувача у фітнес-додатку

```
export const HomePage: FC = () => {
  const user = useCurrentUser();
  const [open, setOpen] = useState(false);
  const drills = user.drills.filter(d => d.date >= startOfWeek(new Date(), { weekStartsOn: 1 }));
  return (
    <>
      {open && <UpdateWeightModal open={open} onClose={() => setOpen(false)} />}

      <Box sx={{ pl: 4, pb: 4 }}>
        <Stack direction="row" gap={3} justifyContent="space-between">
          <WeightChartCard
            weightHistory={user.weightHistory}
            currentWeight={user.weight}
            onNewRecordClick={() => setOpen(true)}
          />
          <FatPercentageChartCard
fatPercentageHistory={user.fatPercentageHistory}
            currentFatPercentage={user.fatPercentage}
            onNewRecordClick={() => setOpen(true)}
          />
        </Stack>
```

Продовження лістингу 3.12

```
            {drills.length === 0 && (
              <Typography level="body-lg" sx={{ mt: 4,
textAlign: "center" }}>
                You have no drills scheduled. Add some to get
started!
              </Typography>
            )}
            <DrillsList drills={drills} />
          </Box>
        </>
      );
    };
```

У лістингу 3.13 показано реалізацію модального вікна, що дозволяє користувачу згенерувати персоналізований план тренувань. Інтерфейс включає поля для введення цільових м'язів та кількості тренувальних днів на тиждень, а також кнопки для підтвердження або скасування генерації.

Лістинг 3.13 – Програмний код, що реалізує модальне вікно генерації персоналізованого тренувального плану

```
    export const GenerateDrillsModal:
FC<GenerateDrillsModalProps> = ({ open, onClose, onSubmit }) =>
      const [generateDrills, { isLoading, isError }] =
useGenerateDrillsMutation();

      const {
        control,
        handleSubmit,
        formState: { errors },
      } = useForm<FormState>({
        resolver: zodResolver(schema),
        defaultValues: { muscles: "", exercisesPerWeek: 0 },
      });
```

Продовження лістингу 3.13

```
              </Stack>
              <Box sx={{ display: "flex", justifyContent:
"space-between", mt: 2 }}>
                <Button type="button" onClick={onClose}
variant="outlined">
                  Cancel
                </Button>
                <Button type="submit" loading={isLoading}>
                  Generate
                </Button>
              </Box>
            </Box>
          </ModalDialog>
        </Modal>
      );
    };
```

У процесі взаємодії користувача з вебдодатком після генерації плану тренувань необхідно надати можливість зберегти його. Для цього реалізовано окреме модальне вікно, яке відкривається після підтвердження створення плану, де користувач може обрати тривалість у тижнях та вказати, чи потрібно оновлювати поточний тиждень – така логіка реалізована у лістингу 3.14.

Важливою особливістю цієї компоненти є перевірка коректності даних, які надсилаються, а також перевірка значення полів imageUrl та link, що дозволяє уникнути передачі невалідних посилань у плані. Після підтвердження дані трансформуються у відповідний формат та надсилаються до API методом applyDrillsPlan, після чого додаток оновлює інформацію про користувача.

Лістинг 3.14 – Програмний код, що реалізує модальне вікно для підтвердження збереження згенерованого плану тренувань

```
export const ApplyDrillsPlanModal: FC<ApplyDrillsPlanModalProps> = ({
  open,
  onClose,
  onSubmit,
  plan,
}) => {
  const [applyDrillsPlan, { isLoading, isError }] = useApplyDrillsPlanMutation();

  const {
    control,
    handleSubmit,
    watch,
    formState: { errors },
  } = useForm<FormState>({
    resolver: zodResolver(schema),
    defaultValues: { durationWeeks: 2, updateCurrentWeek: false },
  });
  const handleFormSubmit = async ({ durationWeeks, updateCurrentWeek }: FormState) => {
    if (!plan) return;
    const payload = plan.map((drill) => ({
      ...drill,
      day: drill.day.toLowerCase() as Day,
      imageUrl: isValid(drill.imageUrl) ? drill.imageUrl,
      exercises: drill.exercises.map((e) => ({
        ...e,
        link: isValid(e.link) ? e.link : null,
      })),
    }));
```

Компонент, наведений у лістингу 3.15, відповідає за відображення користувацьких тренувань, організованих у тижневі блоки відносно поточного тижня. Групування відбувається на основі дати виконання кожної вправи з використанням допоміжної функції, яка обчислює зсув тижня щодо поточної дати. Кожна група вправ супроводжується заголовком, який вказує на відповідний період, наприклад, поточний тиждень або наступний. При натисканні на конкретну вправу відкривається її детальний опис через компонент UserDrillDetails, що забезпечує зручну навігацію користувача в межах інтерфейсу.

Лістинг 3.15 – Програмний код, що відображає тренувальні вправи користувача, згруповані по тижнях відносно поточного

```
export const DrillsList: FC<DrillsListProps> = ({ drills }) => {
  const [searchParams, setSearchParams] = useSearchParams();
  const selectedDrillId = searchParams.get("drillId");
  const selectedDrill = drills.find((d) => d.id === selectedDrillId) || null;
  const grouped = groupByWeekIndex(drills);
  return (
    <>
      <UserDrillDetails
        drill={selectedDrill}
        open={!!selectedDrill}
        onClose={() => setSearchParams({}, { replace: true}
      <Stack>
        {Object.entries(grouped)
          .sort(([a], [b]) => Number(a) - Number(b))
          .map(([indexStr, weekDrills]) => {
            const index = Number(indexStr);
            const start = startOfWeek(addWeeks(new Date(), index), { weekStartsOn: 1 });
```

Продовження лістингу 3.15

```
                const end = endOfWeek(start, { weekStartsOn:1};
                const label =
                  index === -1
                    ? "Previous week"
                    : index === 0
                    ? "This week"
                    : index === 1
                    ? "Next week"
                    : `${start.toLocaleDateString("en-US", {
month: "short", day: "numeric" })} -
${end.toLocaleDateString("en-US", {
                        month: "short",
                        day: "numeric",
                      })
                    }`;
              })}
          </Stack>
        </>
      );
    };

    function groupByWeekIndex<T extends { date: Date }>(items:
T[]): Record<number, T[]> {
      const base = startOfWeek(new Date(), { weekStartsOn: 1
});
      return items.reduce((acc, item) => {
        const week =
differenceInCalendarWeeks(startOfWeek(item.date, {
weekStartsOn: 1 }), base);
        acc[week] = [...(acc[week] || []), item];
        return acc;
      }, {} as Record<number, T[]>);
    }
```

Компонент, представлений у лістингу 3.16, реалізує інтерфейс бічної панелі, яка відкривається після вибору конкретного дня тренувань у списку. У цьому вікні відображається назва тренування, день тижня та повний перелік вправ із відповідними інструкціями, кількістю підходів, повторень і часом на відпочинок. Для кожної вправи передбачена можливість перегляду відеоінструкції, якщо надано коректне посилання. Завдяки компонуванню інформації у зручному для читання вигляді, користувач отримує чітке уявлення про кожен елемент тренувального дня.

Лістинг 3.16 – Програмний код, що відображає деталі обраного дня тренувань у вигляді бічного вікна з переліком вправ, описами, відео та параметрами (кількість підходів, повторів, відпочинок)

```
    export const DrillDetails: FC<DrillDetailsProps> = ({ open, onClose, drill }) => (
      <Drawer open={open} anchor="right" size="md" onClose={onClose}>
        {drill && (
          <Stack p={4}>
            <Typography level="h3">{drill.label}</Typography>
            <Typography level="body-sm" color="neutral">
    {addDays(startOfWeek(new Date(), { weekStartsOn: 1 }),
    Object.values(Day).findIndex((d) => d === drill.day.toLowerCase())).toLocaleDateString("en-US", { weekday: "long" })}
            </Typography>
            <Divider sx={{ my: 2 }} />
            <Stack gap={2} alignItems="center">
              {drill.exercises.map((ex) => (
                <Card sx={{ width: 480 }} variant="soft">
                  <Typography level="title-lg">{ex.name}</Typography>
                  <Typography level="body-md">{ex.instructions}</Typography>
```

Продовження лістингу 3.16

```
                {isValidUrl(ex.link) && (
                  <AspectRatio             minHeight="120px"
maxHeight="200px">
                    <iframe src={ex.link.replace("watch?v=",
"embed/")} allowFullScreen />
                  </AspectRatio>
                )}
                <CardOverflow variant="soft">
                  <CardContent orientation="horizontal">
                    <Stack                   direction="row"
justifyContent="space-around" width="100%">
                      <Stat title="Sets" value={ex.sets} />
                      <Divider orientation="vertical" />
                      <Stat title="Reps" value={ex.reps} />
                      <Divider orientation="vertical" />
                      <Stat  title="Rest"  value={`${ex.rest}
sec`} />
                    </Stack>
                  </CardContent>
                </CardOverflow>
              </Card>
            ))}
          </Stack>
        </Stack>
      )}
    </Drawer>
  );
```

Загальна реалізація вебдодатку охоплює повний цикл взаємодії користувача з системою – від первинного анкетування й генерації персоналізованих тренувань до збереження прогресу й перегляду деталізованих планів. Архітектура реалізована на основі сучасного стеку технологій, таких як Nest.js, React, Firebase, OpenAI API, та забезпечує

інтеграцію фронтенду з серверною логікою через захищені API. У результаті реалізовано адаптивний, функціональний та масштабований інтерфейс, орієнтований на персоналізовану підтримку користувача у досягненні фітнес-цілей.

3.3 Інтерфейс сервісу

Щоб почати користуватися сервісом майбутній користувач переходить до первинного анкетування, яке дає змогу зібрати необхідні персональні дані для подальшої генерації індивідуального тренувального плану. На цьому етапі користувач заповнює поля зросту, ваги, віку, відсотка жирової тканини, а також обирає стать, фітнес-мету, рівень підготовки і вказує можливі обмеження чи травми. Інтерфейс сторінки анкетування у темній темі (рисунок 3.1). Зліва розміщується сама форма, яка містить всі необхідні поля для введення, а праворуч – декоративне фонове зображенняю

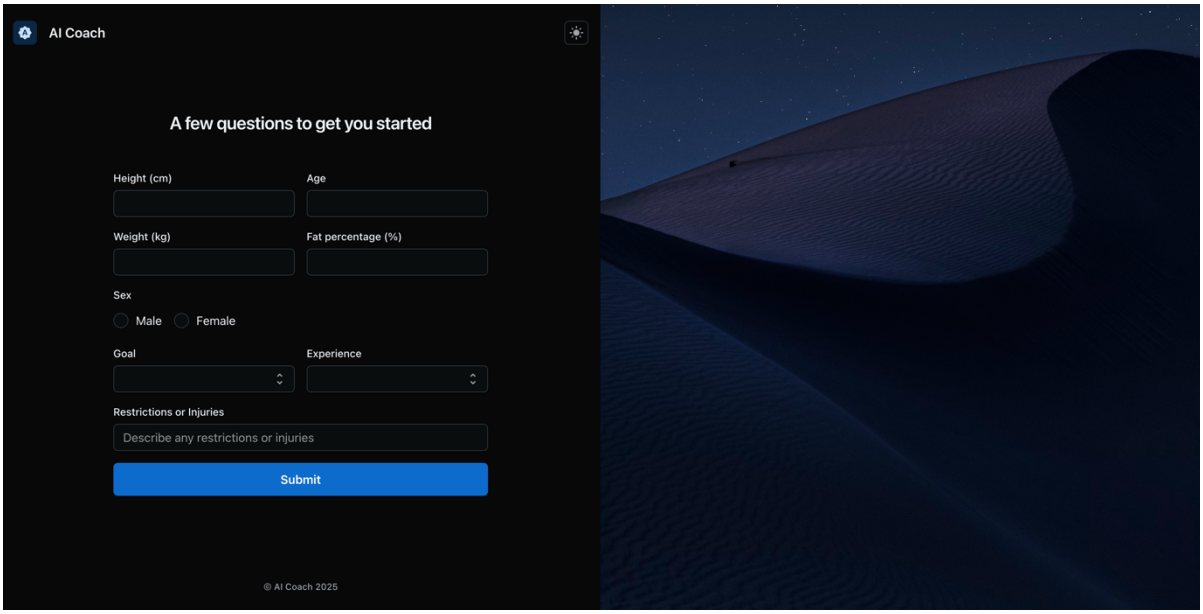

Рисунок 3.1 – Сторінка створення профілю у темній темі

У світлій темі інтерфейс сторінки (рисунок 3.2) анкетування зберігає логіку компонування елементів, забезпечуючи консистентність користувацького досвіду незалежно від обраного режиму. Форма для введення персональних даних розташована зліва, тоді як праву частину займає фонове зображення з альпійським пейзажем, що надає додатку візуальної привабливості. Світле оформлення сприяє кращій видимості полів і тексту, що може бути зручнішим для користувачів у денний час або при роботі на добре освітлених екранах.

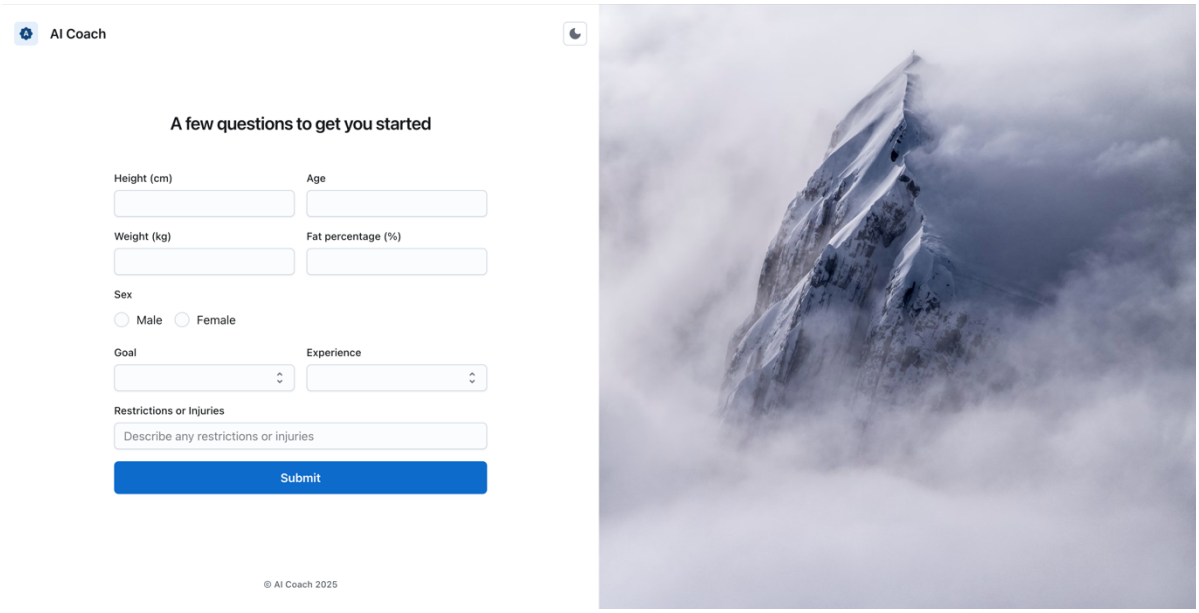

Рисунок 3.2 – Сторінка створення профілю у світлій темі

Далі можна побачити приклад заповненої анкети користувача, яка включає значення основних антропометричних показників, рівень підготовки, ціль тренувань та медичні обмеження (рисунок 3.3). У полі height вказано 176 см, weight – 95 кг, fat percentage – 18 %, а вік користувача становить 21 рік. Обрано стать male, рівень досвіду intermediate, а основною метою визначено gain muscle. Додатково зазначено обмеження у вигляді рекомендації не напружувати область живота.

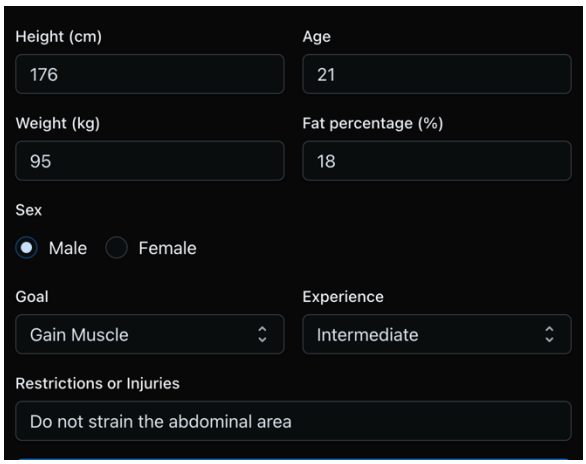

Рисунок 3.3 – Приклад заповненої анкети профілю

Можна побачити головну сторінку користувача, яка реалізована у темній темі й містить основну інформацію про поточну масу тіла та відсоток жиру з графічним відображенням змін за останній рік (рисунок 3.4). У верхній частині інтерфейсу розміщено діаграми з можливістю фіксації нових рекордів, що дозволяє відстежувати динаміку прогресу. Нижче розташовано блок із переліком запланованих тренувань на поточний тиждень, які структуровані за днями.

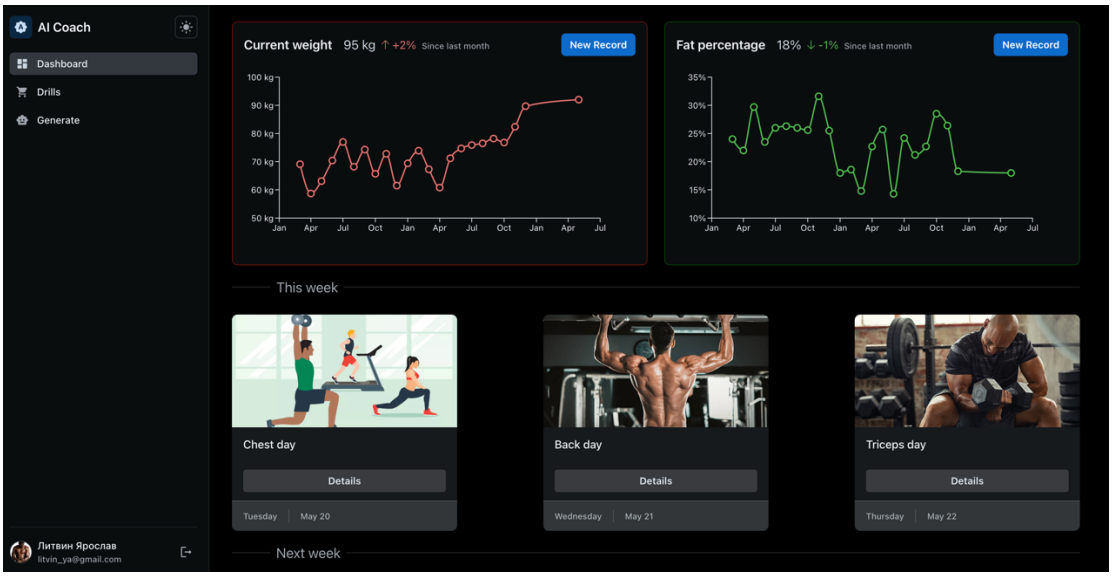

Рисунок 3.4 – Головна сторінка сервісу у темній темі

Далі можна побачити головну сторінку вебдодатку, оформлену у світлій темі, яка забезпечує користувачу доступ до ключової інформації щодо його фізичних параметрів та плану тренувань (рисунок 3.5). Графіки динаміки ваги та відсотка жиру зберігають функціональність і наочність, що дозволяє швидко оцінити прогрес за останні місяці. Інтерактивна кнопка «New Record» на кожному з графіків дозволяє додавати нові показники, зберігаючи історію змін.

Під графіками знаходиться розділ з активними тренуваннями на поточний тиждень, кожне з яких представлено у вигляді картки з назвою, зображенням та кнопкою для перегляду деталей. Користувач може бачити розклад тренувань за днями тижня, що полегшує планування фізичної активності. Світла тема оформлення створює враження простоти та легкості, що сприяє зручності користування додатком.

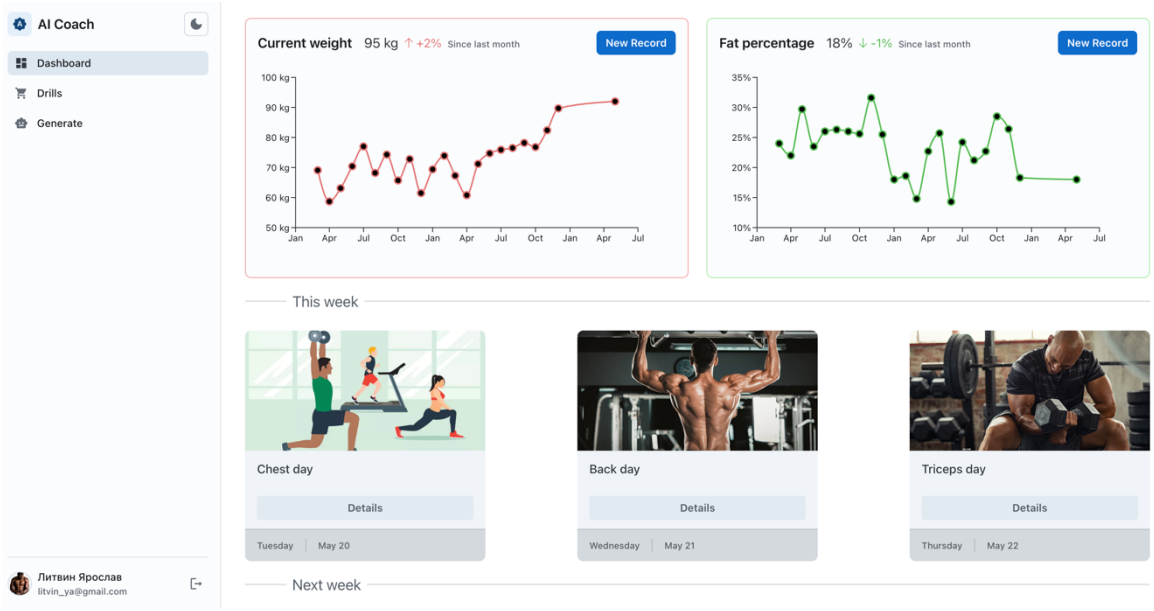

Рисунок 3.5 – Головна сторінка сервісу у світлій темі

Далі можна побачити візуальне сповіщення, яке з’являється у користувача в день запланованого тренування (рисунок 3.6). Інтерфейс реалізовано у вигляді спливаючого вікна з повідомленням про поточне

тренування, яке супроводжується коротким нагадуванням та кнопкою взаємодії. Такий формат повідомлення не лише привертає увагу користувача, а й дає змогу оперативно позначити тренування як завершене.

Використання кольорового маркування та кнопки з підписом Mark as finished підвищує інтуїтивність взаємодії та мотивацію до виконання фізичної активності. Наявність кнопки закриття у правому верхньому куті дозволяє приховати нагадування у разі необхідності, не перериваючи загальний користувацький досвід. Така реалізація сповіщень сприяє підтримці регулярності тренувань та дозволяє користувачу легко вести облік активності.

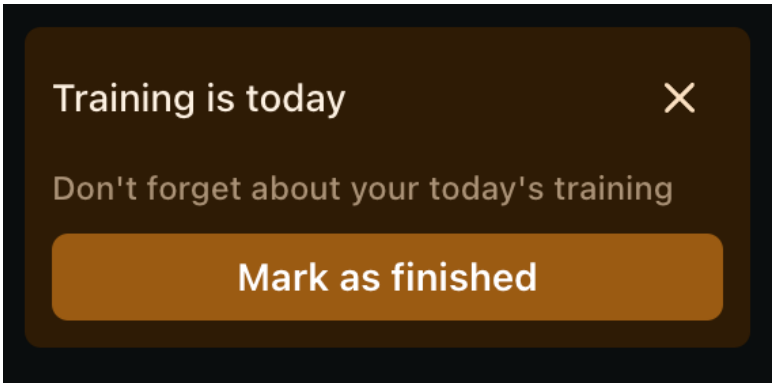

Рисунок 3.6 – Сповіщення про сьогоднішнє тренування

Далі можна побачити ліву панель навігації, яка забезпечує швидкий доступ до основних розділів додатку (рисунок 3.7). У навігаційному меню передбачено пункти Dashboard, Drills та Generate, що відповідають за головну сторінку, список тренувань і генерацію персонального плану відповідно. Активний пункт меню візуально виділений для покращення орієнтації користувача. У верхній частині панелі розміщено кнопку перемикання темного та світлого режиму, яка позначена відповідною іконкою. Це дозволяє користувачу змінити візуальне оформлення інтерфейсу відповідно до індивідуальних уподобань.

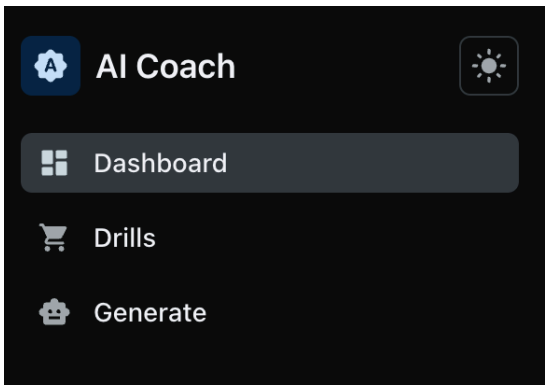

Рисунок 3.7 – Навігаційне меню та кнопка-перемикач теми

Далі можна побачити компонент інтерфейсу, що відображає облікові дані користувача, зокрема ім’я, електронну адресу та аватар (рисунок 3.8). Розміщення цієї панелі у нижній частині навігаційного меню забезпечує швидкий візуальний доступ до інформації про поточну сесію, а також дозволяє виконати вихід із системи за допомогою спеціальної іконки.

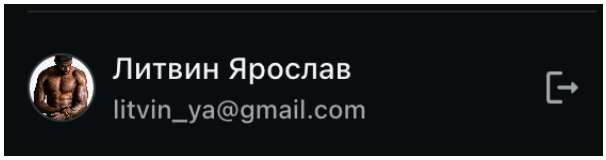

Рисунок 3.8 – Профіль користувача

Графіки зміни ваги тіла та проценту жиру користувача в динаміці за останній рік (рисунок 3.9). Кожна з візуалізацій має підпис, поточне значення, індикатор зміни порівняно з минулим місяцем, а також можливість додати нове значення через кнопку «New Record».

Така реалізація дозволяє користувачеві швидко оцінити прогрес у досягненні фітнес-цілей. Завдяки візуальній подачі даних користувач може відстежувати як позитивні, так і негативні тенденції, що сприяє кращому розумінню впливу тренувального плану на тіло.

Рисунок 3.9 – Трекери ваги та проценту жиру

Користувач має можливість вручну оновити свої показники ваги та відсотка жиру за допомогою компактної форми, яка з’являється у вигляді модального вікна (рисунок 3.10). У відповідні поля вводяться значення у форматі, запропонованому як приклад, після чого натискання кнопки «Add Data» зберігає їх у системі.

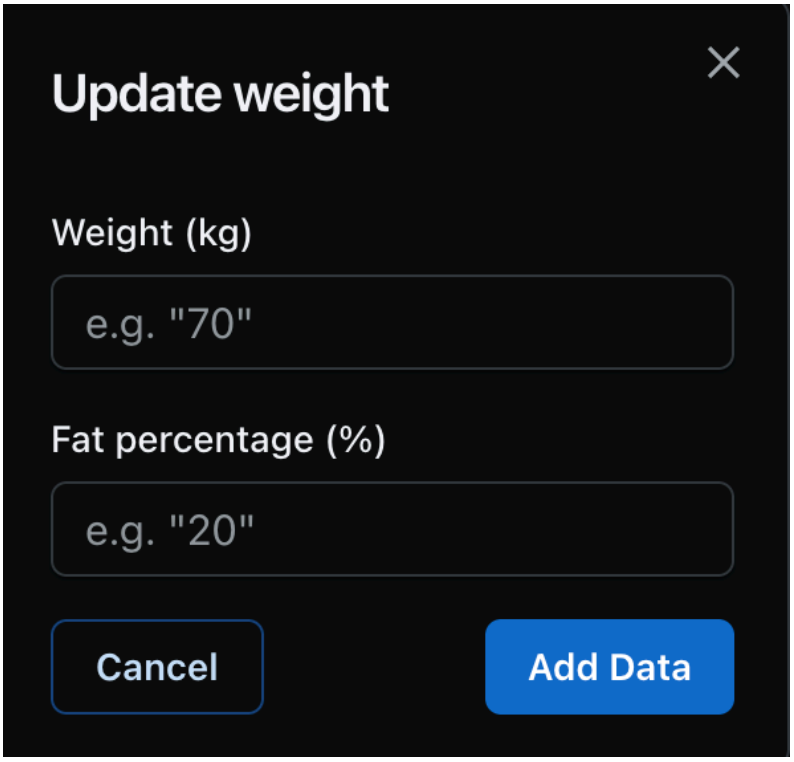

Рисунок 3.10 – Форма оновлення ваги та відсотка жиру користувача

Тепер розглянемо картки тренувань. На першому зображенні зображено картку тренування з назвою chest day, що активна саме сьогодні. Користувач має змогу переглянути деталі тренування або одразу позначити його як виконане, натиснувши на відповідну кнопку.

Дата тренування представлена у нижній частині картки, де також виокремлено мітку today для акцентування на його актуальності. Такий візуальний підхід дозволяє швидко і зручно орієнтуватися у розкладі, стимулюючи до виконання запланованої активності саме в день її призначення (рисунок 3.11).

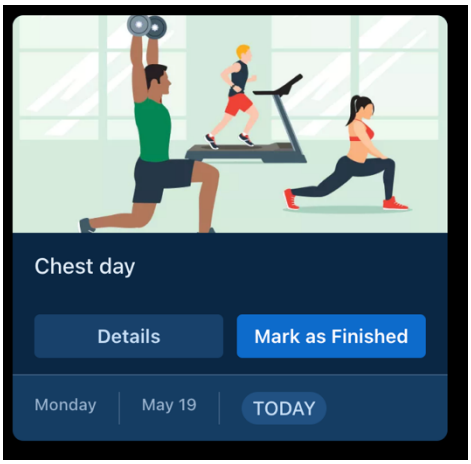

Рисунок 3.11 – Карточка тренування, яка зображує сьогоднішнє завдання

Друге зображення демонструє ту ж картку після того, як тренування було відзначено як завершене. Весь інтерфейс зазнає стилістичних змін: кольорова гамма змінюється на зелений відтінок, що асоціюється з успішністю та виконанням завдання. Зникає кнопка позначення, залишаючи лише інформацію про дату та назву тренування, а також мітку finished. Такий стан інтерфейсу підсилює почуття досягнення і мотиваційно підкріплює користувача на подальші активності, візуально підкреслюючи прогрес у рамках загального плану (рисунок 3.12).

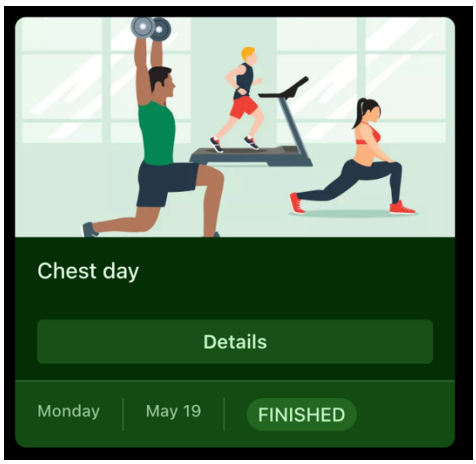

Рисунок 3.12 – Карточка тренування, яка зображує завершене завдання

Третє зображення ілюструє картку тренування back day, яке не було виконане вчасно (рисунок 3.13). Її оформлення кардинально змінено: використовується червоне тонування, що інтуїтивно сигналізує про невиконання або проблему. У нижній частині картки замість today або finished відображається мітка missed, що фіксує факт пропуску. Такий підхід спрямований на інформування користувача про необхідність слідкувати за розкладом, а також слугує підґрунтям для аналітики регулярності тренувань.

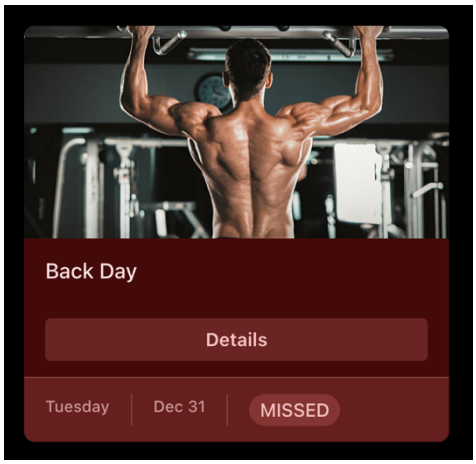

Рисунок 3.13 – Карточка тренування, яка зображує пропущене завдання

На екрані зображено інтерфейс (рисунок 3.14), що дає змогу користувачу переглядати повну історію власних тренувань, згрупованих по тижнях. Кожен блок містить назву тренування, дату проведення, кнопку перегляду деталей та маркер стану: finished, missed або відсутній, якщо тренування ще не відбулося. Колірна індикація використовується для миттєвого візуального розпізнавання: зелений сигналізує про успішне виконання, червоний – про пропущене тренування, а нейтральний фон – про майбутні активності.

Такий підхід дозволяє користувачу швидко орієнтуватися у своєму прогресі та зручніше планувати навантаження. Важливою перевагою є також чітке групування за тижнями, що відповідає загальноприйнятій структурі фітнес-програм. Користувач має змогу оцінити дисципліну та регулярність, а також за необхідності повернутися до деталей кожного з тренувальних днів.

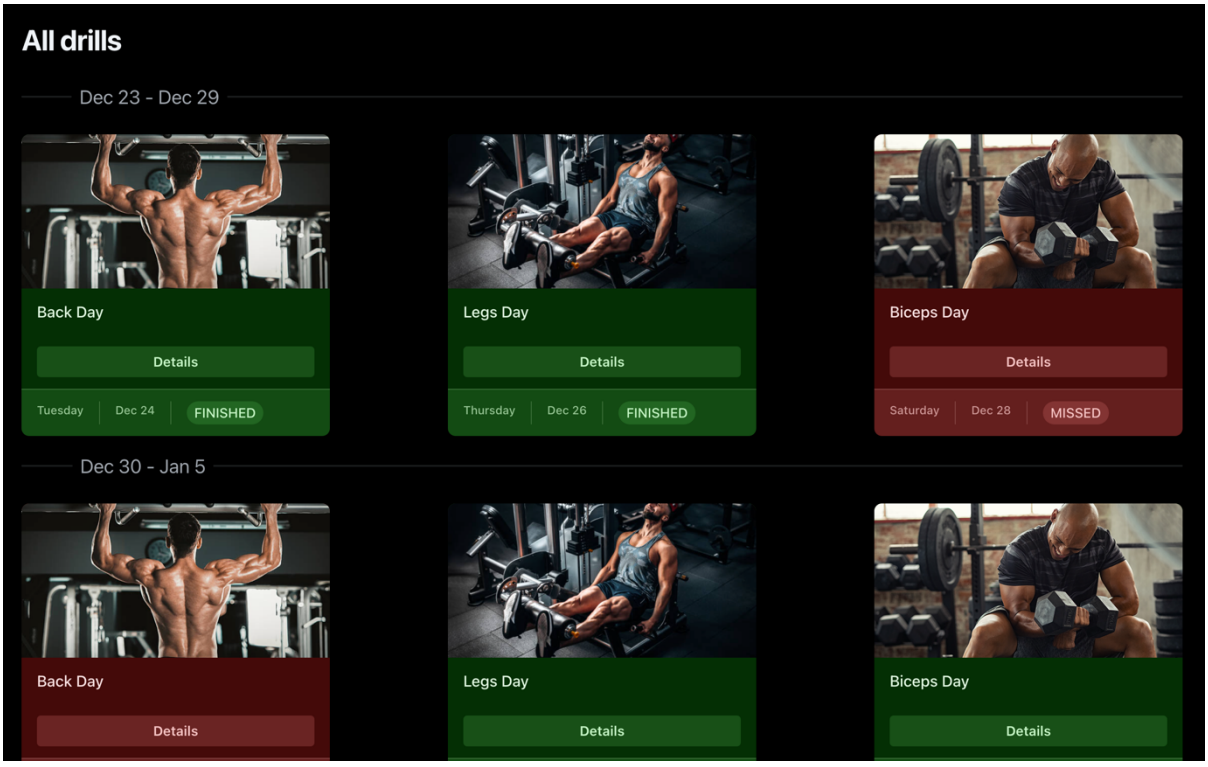

Рисунок 3.14 – Інтерфейс перегляду всіх тренувань користувача

Сторінка генерації тренувального плану відіграє роль точки входу до процесу створення персоналізованої програми вправ. Інтерфейс побудовано таким чином, щоб користувач одразу звернув увагу на основну мету розробленого сервісу – покращення стану здоров'я завдяки індивідуально підібраному фізичному навантаженню. Центральна частина містить мотиваційне повідомлення з акцентом на перевагах платформи, а також заклик до дії у вигляді кнопки «Get started». Далі можна побачити цей інтерфейс, реалізований у темній темі з сучасним дизайном та зображенням у стилі digital-art (рисунок 3.15).

Навігаційне меню, розташоване ліворуч, забезпечує швидкий доступ до основних функціональних розділів вебдодатку – зокрема, дашборду, журналу тренувань та генератора плану. Важливою деталлю є акцент на активному пункті меню, що допомагає користувачу орієнтуватися у структурі інтерфейсу. Загальна композиція сторінки спрямована на створення візуального фокусу, інформативність і легкість у взаємодії, що відповідає сучасним принципам UX-дизайну для фітнес-продуктів.

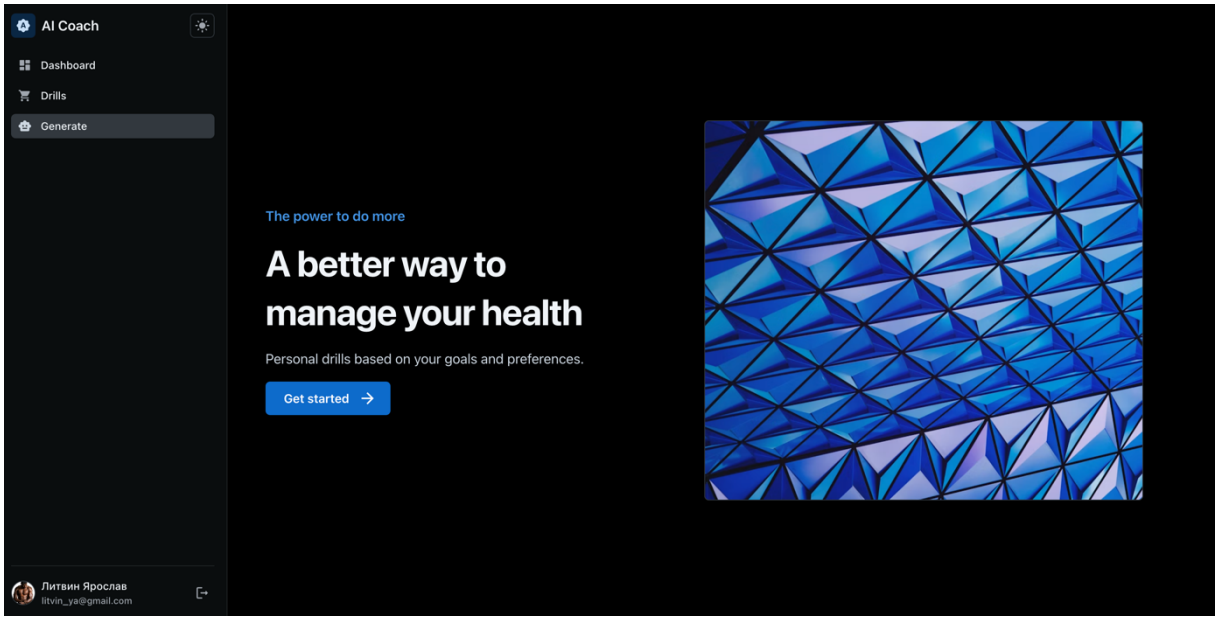

Рисунок 3.15 – Сторінка генерації тренувального плану

Форма генерації плану тренувань дає змогу користувачу самостійно задати цільові групи м’язів та обрати бажану кількість тренувальних днів на тиждень. Введення м’язів реалізовано через текстове поле з підказкою щодо формату, а для вибору кількості днів використано інтуїтивно зрозумілий повзунок. Такий підхід дозволяє гнучко адаптувати подальший план до індивідуальних побажань та навантаження. Далі можна побачити візуалізацію цієї форми у вигляді модального вікна, оформленого у темній темі (рисунок 3.16).

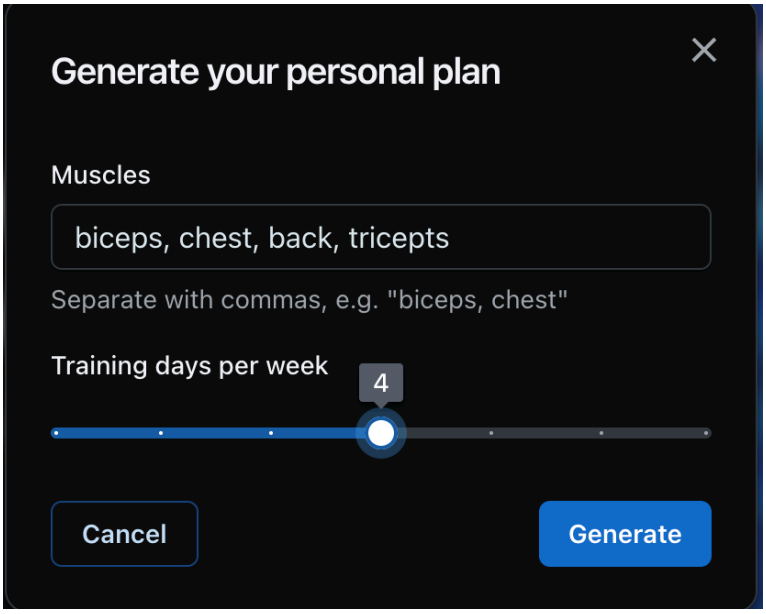

Рисунок 3.16 – Форма генерації плану тренувань

Далі можна побачити інтерфейс попереднього перегляду згенерованого персонального плану тренувань, що включає перелік днів з відповідними тренуваннями у форматі карток (рисунок 3.17). Кожна картка містить назву тренувального дня, ілюстрацію, кнопку переходу до деталей і зазначений день тижня, коли заплановане виконання вправ. Це дозволяє користувачу швидко оцінити структуру створеного плану перед його застосуванням.

У верхній частині праворуч передбачено кнопку Apply, яка виконує функцію підтвердження вибору та збереження плану. Таким чином, користувач отримує змогу адаптувати тренувальний тиждень до власного розкладу ще до фінального затвердження, що підвищує зручність і гнучкість взаємодії з сервісом.

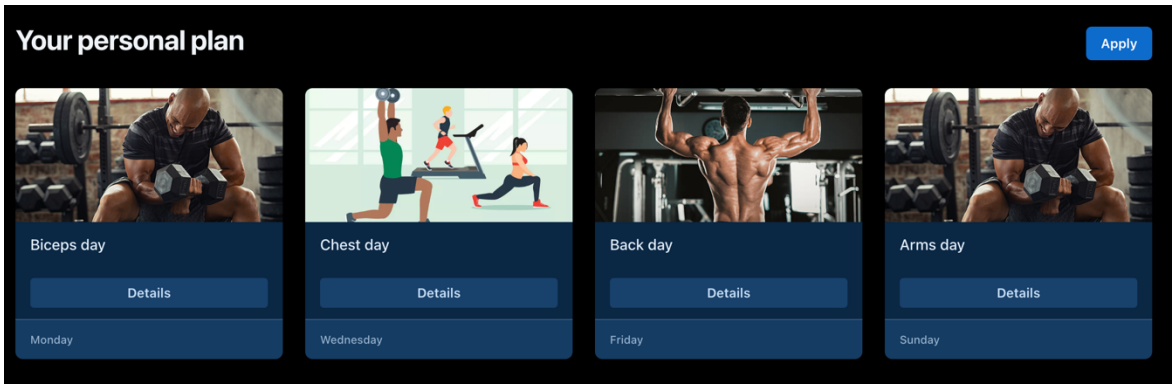

Рисунок 3.17 – Інтерфейс попереднього перегляду згенерованого тренувального плану з кнопкою підтвердження

Інтерфейс деталей обраного тренування відображає всю необхідну інформацію для користувача у форматі карток з вправами (рисунок 3.18). Кожна вправа включає назву, текстову інструкцію, відео з демонстрацією техніки виконання, а також параметри тренування: кількість підходів, повторів і час відпочинку між ними. Такий підхід дозволяє ознайомитись зі змістом без необхідності переходу на сторонні ресурси.

Відеоматеріали інтегровані безпосередньо в інтерфейс і є одним з ключових елементів, що забезпечують візуальну підтримку користувача при виконанні вправ. Структура представлення контенту є однаковою для кожного елемента, що створює відчуття цілісності і спрощує навігацію в межах одного дня. Завдяки цьому користувач може зосередитися на виконанні вправ відповідно до індивідуального плану, не витрачаючи зайвий час на пошук інструкцій.

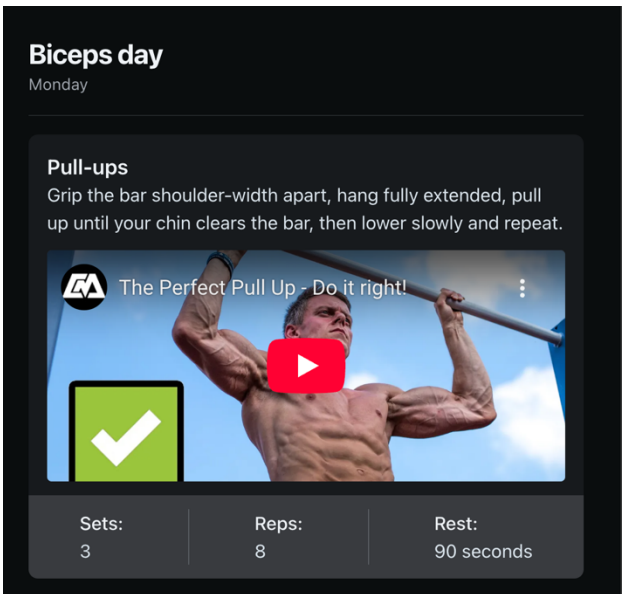

Рисунок 3.18 – Інтерфейс деталей обраного тренування

На основі отриманих даних персонального плану, користувач отримує доступ до інтерфейсу, де представлено тренування із зазначенням назви дня, наприклад back day, а також дати його виконання. Кожна секція тренування містить окремі вправи, що структуровані у вигляді компактних карток з усіма необхідними атрибутами для виконання. Далі можна побачити структуру тренування у день для м'язів спини, яка включає вправи lat pulldown та seated row (рисунок 3.19).

Опис кожної вправи містить чітку покрокову інструкцію, що полегшує сприйняття алгоритму виконання навіть для новачка. Наприклад, для вправи lat pulldown вказано необхідність правильно відрегулювати положення сидіння та тримати хват широким, а потім підтягувати штангу до верхньої частини грудей. У вправі seated row фокус робиться на положенні ніг, утриманні ручки обома руками та повільному поверненні у вихідне положення після завершення тяги. Такі тексти є стислими, але достатньо інформативними для правильного виконання.

Кожна картка доповнена вбудованим відео, що демонструє правильну техніку виконання вправи. Це дає можливість візуально ознайомитися з рухами перед тим, як приступити до практики. Відео інтегруються з

популярних джерел і дозволяють користувачам уникнути помилок, які часто виникають у новачків через неправильну техніку. У поєднанні з текстовим описом та візуалізацією створюється ефективне навчальне середовище.

Під відео міститься зведена інформація про параметри вправ: кількість підходів, повторень і час відпочинку між ними. У випадку з lat pulldown це 4 підходи по 10 повторень із 90 секундами відпочинку. Це дозволяє користувачам швидко зорієнтуватися у структурі тренування та точно дотримуватися рекомендованого навантаження. Таке подання сприяє покращенню користувацького досвіду та підвищує ефективність взаємодії з тренувальною системою.

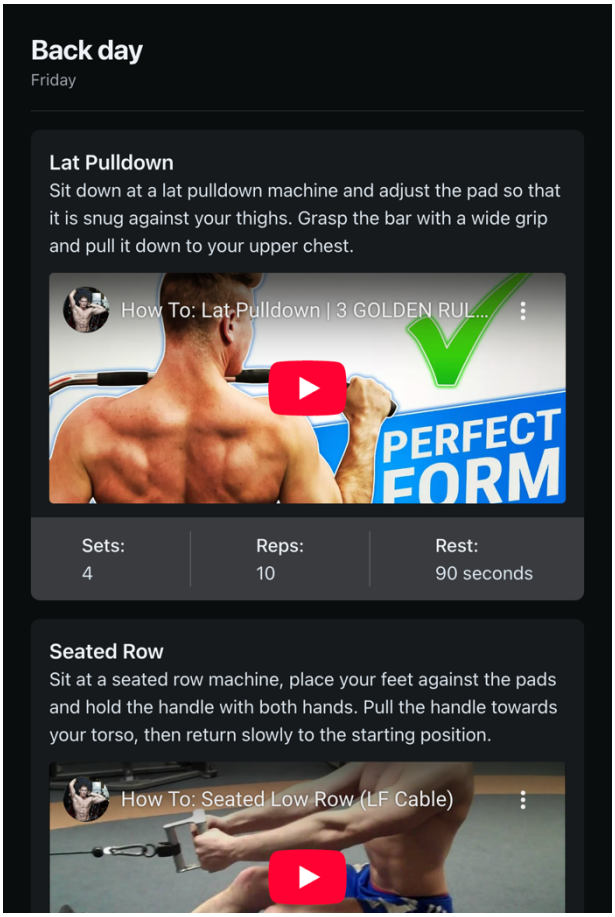

Рисунок 3.19 – Зміст тренування з описом, відео та кількістю підходів з повтореннями

У процесі генерації персоналізованого тренувального плану користувач має можливість вказати бажану тривалість плану у тижнях за допомогою інтерактивного повзунка. Також передбачено перемикач, що дозволяє визначити, чи потрібно оновлювати поточний тиждень, тобто чи слід замінити вже існуючі тренування новими.

Такий підхід забезпечує гнучкість взаємодії з системою і дозволяє адаптувати новий план без втрати актуальності для поточного тижня. Далі можна побачити приклад такого інтерфейсу, представлений у модальному вікні (рисунок 3.20).

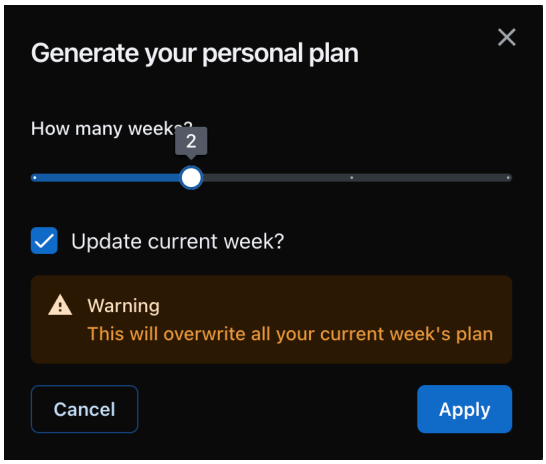

Рисунок 3.20 – Модальне вікно вибору тривалості та оновлення поточного тижня при генерації плану

Особливістю реалізації є наявність попереджувального повідомлення, яке з’являється лише у випадку, коли користувач активує опцію оновлення поточного тижня. Повідомлення чітко інформує про наслідки дії – повне перезаписування вже існуючого плану на обраний тиждень. Це дозволяє уникнути випадкової втрати даних та знижує ризик помилкових дій. Візуальна ієрархія вікна та колірне оформлення зроблені таким чином, щоб ключові елементи були максимально помітними для користувача.

ВИСНОВКИ

У межах кваліфікаційної роботи було повністю виконано поставлене завдання – розроблено персоналізований вебдодаток для генерації індивідуальних планів тренувань із використанням штучного інтелекту. Реалізовано повний цикл функціональності: від первинного анкетування користувача до створення та застосування сесій тренувань, які враховують його фізичні характеристики, рівень підготовки, цілі, а також наявні обмеження чи травми. Додаток забезпечує інтерактивну взаємодію користувача з інтерфейсом, підтримує збереження історії показників (ваги та жиру) та дозволяє перегляд і зміну плану у зручній формі. Візуальна частина побудована із урахуванням вимог адаптивного дизайну, передбачено підтримку темного та світлого режимів для зручності користувача.

Якісними показниками роботи системи є стабільність та надійність функціонування API-запитів, коректна обробка даних та валідація форм, а також швидка генерація персоналізованих планів за допомогою інтеграції з OpenAI API. Кількісні результати тестування показали високу продуктивність рендерингу інтерфейсу навіть при великій кількості тренувальних записів (до 100+), а також мінімальний середній час відповіді бекенду на генерацію плану (менше 1.5 секунди на запит до GPT-4o). Значна увага приділена безпеці – авторизація здійснюється через сервіс Auth0, що забезпечує надійний захист облікових даних користувача. Усі персональні дані зберігаються в хмарній базі Firebase, з доступом через обмежений Firestore SDK.

Порівнюючи розроблену систему з існуючими аналогами, такими як Fitbod, Freeletics або BetterMe, можна стверджувати, що запропоноване рішення має низку переваг. Зокрема, розроблений вебдодаток забезпечує вищу гнучкість у налаштуванні тренувань, оскільки не базується лише на заздалегідь підготовлених шаблонах, а використовує генеративний ШІ для

створення унікального контенту. Система підтримує можливість динамічного оновлення планів та дозволяє адаптувати їх під зміну фізичних параметрів користувача, що є рідкістю для багатьох існуючих додатків. Окрім того, реалізований вебінтерфейс з повною підтримкою контролю історії ваги й жиру, а також підрахунком тренувальних сесій, наближує сервіс до повноцінної системи персонального фітнес-наставника.

На основі проведеної роботи можна рекомендувати декілька напрямків подальшого розвитку. По-перше, доцільно інтегрувати трекери активності (наприклад, через Google Fit або Apple HealthKit), що дозволить автоматично враховувати фізичну активність користувача та відповідно адаптувати план. По-друге, варто реалізувати систему сповіщень, яка буде нагадувати про тренування, або рекомендувати день відпочинку на основі динаміки ваги та рівня виконаних вправ. По-третє, можливим є розширення функціональності за рахунок включення дієтичного супроводу, що формуватиметься на основі тих же параметрів користувача та буде синхронізований із тренувальним планом. Нарешті, у контексті масштабування проєкту можна розглянути розробку мобільної версії на базі React Native для охоплення ширшої аудиторії.

Таким чином, результати виконання кваліфікаційної роботи засвідчують практичну цінність запропонованої розробки та високий потенціал її використання у сфері персоналізованого фітнес-консалтингу. Застосування сучасних технологій, таких як OpenAI API, Firebase та інфраструктура хмарного зберігання, у поєднанні з ретельно продуманим UI/UX рішенням, дозволяє позиціонувати вебдодаток як конкурентоспроможний продукт із широкими можливостями для майбутнього вдосконалення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Auth0. Identity platform documentation. URL: https://auth0.com/docs(date of access: 10.04.2025).

2. FitOn Inc. FitOn: Fitness Workout Plans. URL: https://fitonapp.com(date of access: 04.03.2025).

3. Freeletics GmbH. Freeletics – Personal Trainer App. URL: https://www.freeletics.com(date of access: 17.04.2025).

4. Gamification for health and wellbeing: A systematic review of the literature / D. Johnson et al. *Internet Interventions*. 2016. Vol. 6. P. 89–106. URL: https://doi.org/10.1016/j.invent.2016.10.002(date of access: 08.04.2025).

5. Google Cloud. Firestore documentation. URL: https://cloud.google.com/firestore/docs(date of access: 05.02.2025).

6. Jefit Inc. Jefit Workout Planner Gym Log. URL: https://www.jefit.com(date of access: 22.02.2025).

7. Joy UI. MUI Joy UI documentation. URL: https://mui.com/joy-ui/getting-started/(date of access: 11.04.2025).

8. Kreps G. L., Neuhauser L. New directions in eHealth communication: Opportunities and challenges. *Patient Education and Counseling*. 2010. Vol. 78, no. 3. P. 329–336. URL: https://doi.org/10.1016/j.pec.2010.01.013(date of access: 01.04.2025).

9. Lieberoth A., others. The effectiveness of digital health apps and wearables for weight loss: Meta-analysis of randomized controlled trials. *Journal of Medical Internet Research*. 2021. Vol. 23, no. 7. P. 27251. URL: https://doi.org/10.2196/27251(date of access: 25.04.2025).

10. MyFitnessPal, Inc. MyFitnessPal. URL: https://www.myfitnesspal.com(date of access: 21.04.2025).

11. NestJS. A progressive Node.js framework. URL: https://docs.nestjs.com(date of access: 15.04.2025).

12. Nike, Inc. Nike Training Club App. URL: https://www.nike.com/ntc-app(date of access: 16.04.2025).

13. OpenAI. API documentation. URL: https://platform.openai.com/docs(date of access: 15.04.2025).

14. Statista. Fitness app market revenue worldwide 2017–2027. URL: https://www.statista.com/statistics/1120195/fitness-app-market-revenue-worldwide/(date of access: 18.04.2025).

15. The rise of consumer health wearables: Promises and barriers / L. Piwek et al. *PLoS Medicine*. 2016. Vol. 13, no. 2. P. 1001953. URL: https://doi.org/10.1371/journal.pmed.1001953(date of access: 28.03.2025).

16. Topol E. Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again. Basic Books, 2019.

17. World Health Organization. Global action plan on physical activity 2018–2030: More active people for a healthier world. URL: https://www.who.int/publications/i/item/9789241514187(date of access: 23.03.2025).

18. World Health Organization. Physical activity. URL: https://www.who.int/news-room/fact-sheets/detail/physical-activity(date of access: 19.03.2025).