МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Навчально-науковий інститут інформаційних технологій та бізнесу
Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавра

на тему: «**Розробка інтелектуальної експертної системи для автоматизованого планування та моніторингу фізичного розвитку користувачів»**

Виконав: студент 4 курсу, групи КН-41
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Роман Володимирович Давидюк

Керівник: *викладач, фахівець практик Сергій Васильович Ляховчук*

Рецензент: *кандидат технічних наук, доцент, доцент кафедри прикладної математики Донецького національного університету імені Василя Стуса*
Загоруйко Любов Василівна

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

Завідувач кафедри інформаційних технологій та аналітики даних
_____ (проф., д.е.н. Кривицька О.Р.)

Протокол № 11 від « 20 » травня 2026 р

Острог, 2026

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня бакалавра

Тема: *Розробка інтелектуальної експертної системи для автоматизованого планування та моніторингу фізичного розвитку користувачів*

Автор: *студент спеціальності 122 Комп'ютерні науки освітньо-професійної програми «Комп'ютерні науки»*
Давидюк Роман Володимирович

Науковий керівник: *викладач, фахівець практик*
Сергій Васильович Ляховчук

Захищена «……»…… 2026 року.

Пояснювальна записка до кваліфікаційної роботи: ***72*** *(кількість сторінок роботи) с.,* ***16*** *(кількість рисунків) рис.,* ***0*** *(кількість таблиць) табл.,* ***0*** *(кількість додатків) додатків,* ***43*** *(кількість джерел) джерел.*

.

Ключові слова: *ІНТЕЛЕКТУАЛЬНА ЕКСПЕРТНА СИСТЕМА, ПЕРСОНАЛІЗОВАНИЙ ФІТНЕС, НУТРИЦІОЛОГІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, .NET 10, REACT, CQRS, CLEAN ARCHITECTURE, КОМП'ЮТЕРНИЙ ЗІР, AI FOOD SCANNER.*

Короткий зміст праці: *Кваліфікаційна робота присвячена проєктуванню та розробці інтелектуальної експертної системи «Vector» для автоматизованого планування персоналізованих програм тренувань і харчування, а також моніторингу фізичного стану користувачів. Метою роботи є створення комплексного вебзастосунку, що інтегрує сучасні технології штучного інтелекту для забезпечення глибокої індивідуалізації фітнес-процесу. У процесі дослідження обґрунтовано вибір технологічного стека (.NET 10, React 18, PostgreSQL) та реалізовано архітектуру на базі Onion Architecture із застосуванням патерну CQRS для розділення операцій читання та запису. Програмно реалізовано інтеграцію з Gemini API для динамічної генерації тренувальних планів та модулями комп'ютерного зору (Vision AI) для автоматичного розпізнавання харчової цінності страв за фотографіями. Клієнтська частина спроєктована за принципом Mobile-First з використанням Tailwind CSS, Redux Toolkit та Framer Motion для створення адаптивного*

ABSTRACT
of the qualification
work for a Bachelor's degree

Theme: *Development of an intelligent expert system for automated planning and monitoring of users' physical development*

Author: *Student of degree programme 122 Computer Science within the vocational education programme «Computer Science»*
Roman Volodymyrovych Davydiuk

Supervisor: *Lecturer, Practising Specialist*
Serhii Vasyliovych Liakhovchuk

Defended on «……»…… 2026.

Explanatory note to the qualification work: ***72*** *(number of pages) pp.,* ***16*** *(number of figures) figs., 0 (number of tables) tables, 0 (number of appendices) appendices,* ***43*** *(number of sources) sources.*

Keywords: *INTELLIGENT EXPERT SYSTEM, PERSONALIZED FITNESS, NUTRITION, ARTIFICIAL INTELLIGENCE, .NET 10, REACT, CQRS, CLEAN ARCHITECTURE, COMPUTER VISION, AI FOOD SCANNER.*

Abstract: *This qualification work is dedicated to the design and development of the "Vector" intelligent expert system for automated planning of personalized workout and nutrition programs, as well as monitoring users' physical progress. The aim of the work is to create a comprehensive web application that integrates modern artificial intelligence technologies to ensure deep individualization of the fitness process. During the research, the technology stack (.NET 10, React 18, PostgreSQL) was justified, and the architecture was implemented based on Onion Architecture using the CQRS pattern to separate read and write operations. The integration with Gemini API for dynamic generation of workout plans and computer vision modules (Vision AI) for automatic recognition of the nutritional value of meals from photographs was programmatically implemented. The frontend is designed following the Mobile-First principle using Tailwind CSS, Redux Toolkit, and Framer Motion to create a responsive interface. The software product is a complete platform ready for deployment in a cloud environment.*

Зміст

Вступ

Актуальність теми. У сучасних умовах стрімкого розвитку технологій та зростання уваги суспільства до здорового способу життя питання ефективного планування фізичного розвитку та харчування набуває особливої актуальності. Сидячий спосіб життя, неправильне харчування та відсутність індивідуального підходу до тренувань призводять до погіршення здоров'я населення, збільшення випадків ожиріння, серцево-судинних захворювань та зниження загального рівня фізичної підготовки.

Традиційні фітнес-програми та додатки, як правило, пропонують уніфіковані рішення, які не враховують індивідуальні особливості користувача: вік, стать, рівень підготовки, стан здоров'я, доступне обладнання, спосіб життя та особисті цілі. У зв'язку з цим виникає потреба у розробці інтелектуальних систем, здатних автоматично генерувати персоналізовані плани тренувань і харчування, а також здійснювати постійний моніторинг прогресу користувача.

Використання штучного інтелекту у сфері фітнесу та нутриціології дозволяє значно підвищити ефективність занять, зробити їх доступнішими та персоналізованішими. Саме тому розробка інтелектуальної експертної системи для автоматизованого планування та моніторингу фізичного розвитку користувачів є актуальним і перспективним напрямком.

Мета роботи. Метою кваліфікаційної роботи є розробка інтелектуальної експертної системи для автоматизованого планування персоналізованих програм тренувань і харчування, а також моніторингу фізичного розвитку користувачів.

Об'єкт дослідження – процес автоматизованого планування та моніторингу фізичного розвитку користувачів за допомогою програмних засобів.

Предмет дослідження – інструментальні засоби та технології розробки інтелектуальної системи, зокрема архітектура програмного забезпечення, методи штучного інтелекту, веб-технології та інтеграція компонентів для створення ефективної експертної системи.

Завдання роботи:

- провести аналіз предметної області персонального фітнесу, харчування та існуючих програмних рішень;
- спроектувати архітектуру інтелектуальної експертної системи;
- розробити функціональні модулі для генерації персоналізованих тренувальних та харчових планів;
- реалізувати систему моніторингу виконання планів, включаючи аналіз фото прийомів їжі;
- здійснити вибір сучасного інструментарію розробки та тестування програмного продукту;
- розробити та впровадити користувацький інтерфейс веб-додатка.

Методи дослідження.У роботі використано методи системного аналізу предметної області, методи об’єктно-орієнтованого та компонентного проектування програмного забезпечення, технології штучного інтелекту для генерації рекомендацій, а також методи програмної реалізації та тестування веб-додатків.

Розділ 1. Загальні положення

1.1. Опис предметного середовища

В умовах сьогодення спостерігається інтенсивний технологічний прогрес, швидка урбанізація та глобальні зміни у звичках людей. Паралельно з цим фіксується суттєве зниження загального рівня здоров'я, що зумовлено браком фізичної активності, незбалансованим раціоном, постійними стресами та ігноруванням комплексних тренувань. Згідно зі статистикою ВООЗ, проблема зайвої ваги стосується більш ніж 1,9 мільярда дорослого населення планети, причому 650 мільйонів із них мають діагноз ожиріння. Саме гіподинамія виступає ключовим тригером для виникнення хвороб серцево-судинної системи, діабету другого типу, ракових пухлин і призводить до ранньої смертності.

Сучасна фітнес-індустрія об'єднує різноманітні методики, покликані вдосконалювати фізичний стан тіла, підвищувати силу, витривалість, гнучкість і загальну координацію рухів. Сьогодні ця сфера еволюціонувала від звичайних занять спортом до комплексної дисципліни, невіддільної від нутриціології — науки, що вивчає вплив продуктів харчування та метаболічних процесів на здоров'я.

Індивідуальний підхід до фізичного розвитку вимагає розробки унікальних тренувальних і дієтичних планів, що базуються на численних параметрах користувача: вікових і статевих ознаках, антропометрії, поточній спортивній формі, медичних протипоказаннях, наявності інвентарю, розпорядку дня та конкретних цілях (від схуднення до набору маси), враховуючи при цьому смакові преференції та психотип особистості.

Проте більшість представлених на ринку програмних продуктів не позбавлені серйозних вад. Стандартні алгоритми мобільних застосунків чи рекомендації у спортзалах часто базуються на загальних шаблонах, які не вміють гнучко підлаштовуватися під динаміку змін в організмі клієнта. З іншого боку, калькулятори калорій переважно вимагають ручного внесення спожитих продуктів, що забирає багато часу і підвищує ймовірність помилок. До того ж, на ринку майже відсутні

комплексні платформи, які б органічно поєднували створення спортивних програм, розумний трекінг раціону та безперервне відстеження результатів.

Саме тому впровадження технологій штучного інтелекту в індустрію здоров'я стає вкрай необхідним кроком. Новітні нейромережі здатні автоматично розпізнавати склад їжі за фотознімками, вираховувати добові норми мікро- та макроелементів, а також формувати оптимальні дієтичні поради. Симбіоз такого AI-аналізу з традиційними фітнес-алгоритмами дає змогу проєктувати потужні експертні платформи нового покоління.

Варто підкреслити, що діджиталізація велнес-сектору є одним із найпотужніших світових трендів сьогодення. Аналітика Grand View Research свідчить, що станом на 2024 рік обсяг ринку застосунків для фітнесу перетнув позначку в 15 мільярдів доларів і демонструє стрімку динаміку до розширення. Користувачі масово відмовляються від універсальних методик на користь індивідуалізованих сервісів, усвідомлюючи, що довгостроковий успіх можливий лише за умови врахування особистих фізіологічних нюансів.

Найвищу ефективність демонструє саме всебічний підхід, де гармонійно поєднані силові вправи, кардіо, якісне відновлення та збалансоване меню. Грамотний раціон здатен покращити спортивні результати на 30–40 %, тоді як ігнорування дієти нівелює старання навіть за умов систематичних походів у зал.

Понад те, сьогодні юзери шукають не базові калькулятори чи каталоги вправ, а просунуті системи, які вміють:

- самостійно коригувати навантаження відповідно до нових досягнень;
- брати до уваги ступінь фізичної втоми та швидкість відновлення;
- проводити аналіз спожитих страв за допомогою комп'ютерного зору;
- забезпечувати якісну підтримку та зрозумілі підказки;
- взаємодіяти з людиною її рідною мовою.

Для українського суспільства питання якісного персонального супроводу у спорті та харчуванні є надзвичайно актуальним. Гіподинамія, дефіцит фахівців у регіонах та високі ціни на послуги професійних тренерів формують гостру потребу у доступному цифровому AI-інструменті. З огляду на це, проєктування розумної

платформи, що базується на нейромережах, глибокій персоналізації та має ергономічний дизайн, перетворюється на пріоритетне завдання з великою соціальною та комерційною цінністю.

Отже, предметна область розроблюваного застосунку охоплює наступні компоненти:

- механізми побудови тренувальних циклів з огляду на унікальні риси організму;
- алгоритми обчислення калоражу та балансу макронутрієнтів (БЖВК);
- інструментарій для контролю за дотриманням спортивних і харчових графіків;
- застосування інноваційних ІТ-рішень для повної автоматизації вищезгаданих задач.

Створення інтелектуального програмного комплексу, орієнтованого на одночасне розв'язання проблем персоналізації тренінгу, розумного харчування та безперервного моніторингу здоров'я, виступає закономірним та багатообіцяючим етапом цифрової трансформації Health & Fitness індустрії.

1.2. Огляд існуючих аналогів

Сьогодні індустрія цифрових продуктів для здорового способу життя та дієтології переповнена різноманітними вебсервісами та мобільними програмами. Щоб чітко визначити конкурентні переваги нашої системи, було здійснено огляд ключових гравців на ринку, яких доцільно класифікувати за трьома категоріями: багатофункціональні спортивні трекери, вузькоспрямовані програми для контролю раціону та новітні сервіси на базі штучного інтелекту.

Глобальним флагманом у сегменті відстеження калорійності та балансу нутрієнтів виступає сервіс MyFitnessPal. Цей продукт вирізняється ергономічним дизайном, підтримкою розпізнавання штрих-кодів та колосальною бібліотекою харчових продуктів. Втім, його ключовим мінусом залишається неспроможність формувати повноцінні індивідуальні графіки занять; платформа працює скоріше як щоденник, аніж як генератор комплексних стратегій фізичного розвитку.

Такі застосунки, як Apple Fitness+ та Nike Training Club, надають доступ до широкого асортименту заздалегідь створених відеотренувань. Попри бездоганну візуальну складову та потужні мотиваційні інструменти, їхні методики носять стандартизований характер і фактично ігнорують унікальні параметри організму, наявний у клієнта інвентар чи його актуальну фізичну форму.

Платформи Strong та Hevy орієнтовані виключно на силовий тренінг і функціонують як електронні журнали. Хоча вони пропонують комфортний інструментарій для запису підходів та ваг, ці програми не інтегрують дієтичний супровід і позбавлені функції автоматичного створення спортивних програм.

У сегменті продуктів, що активно залучають ШІ, слід відзначити Noom, Dr. Muscle, FitnessAI та Future. Зокрема, Future просувається на ринку як цифровий наставник, формуючи унікальні плани, які згодом підлягають ручному коригуванню живим фахівцем. Слабостями цього підходу є значна ціна послуг та неминуча прив'язка до роботи реальної людини. Натомість сервіс Noom робить ставку на психологічні аспекти харчування, використовуючи методи когнітивно-поведінкової терапії. Ця програма чудово справляється з корекцією харчових звичок, але майже повністю ігнорує розробку розгорнутих планів силових навантажень. Додатки на кшталт Boostcamp та Alpha Progression здатні алгоритмічно проєктувати силові цикли, проте залишають поза увагою питання дієтології та всебічного відстеження змін у тілі.

Вітчизняний ринок у цій галузі перебуває на стадії становлення: українські продукти зазвичай обмежуються функціями примітивних лічильників калорій або є просто локалізованими клонами іноземних сервісів, не забезпечуючи серйозної персоналізації чи впровадження нейромереж.

Зіставлення наявних програмних продуктів демонструє низку критичних прогалин:

- відсутність єдиної екосистеми, що об'єднує спорт, дієту та аналітику;
- поверхнева індивідуалізація під потреби конкретного юзера;
- ігнорування інструментів комп'ютерного зору для розпізнавання страв на фотографіях;

- нездатність графіків тренувань швидко підлаштовуватися під актуальний прогрес чи втому людини;
- недостатня або повністю відсутня україномовна локалізація.

З огляду на це, формується чіткий запит на проєктування експертної системи нового зразка, яка б гармонійно поєднала генеративний ШІ для створення унікальних спортивних і дієтичних стратегій, автоматизований візуальний аналіз їжі, безперервний трекінг результатів та інтуїтивний дизайн із повноцінним українським інтерфейсом.

1.3 Постановка задачі та мета розробки

Дослідження ринку цифрових продуктів та аналіз специфіки обраної галузі засвідчили, що попри розмаїття фітнес-застосунків, наразі спостерігається дефіцит єдиної багатофункціональної платформи, яка б органічно поєднувала новітні алгоритми штучного інтелекту, максимальну адаптивність до користувача та тісну інтеграцію спортивного планування з дієтологічним супроводом.

Головне завдання цієї кваліфікаційної роботи полягає у проєктуванні та практичній реалізації інтелектуального програмного комплексу. Він має автоматизувати процеси створення унікальних спортивних програм, генерувати індивідуальні плани харчування та забезпечувати безперервний контроль за фізичним прогресом людини.

Щоб успішно втілити цей задум у життя, необхідно виконати низку послідовних кроків:

1. Здійснити ґрунтовний огляд сфери нутриціології, персонального тренінгу та проаналізувати наявні IT-продукти в цьому сегменті.
2. Сформувати перелік як функціональних, так і системних (нефункціональних) вимог до майбутньої платформи.
3. Розробити загальну архітектуру програмного продукту та спроєктувати структуру реляційної бази даних.
4. Створити серверне ядро (Backend) застосунку, імплементуючи сучасні патерни проєктування, такі як Onion Architecture та CQRS.

5. Побудувати клієнтський інтерфейс (Frontend), гарантуючи його сучасний вигляд та високий рівень ергономічності.
6. Інтегрувати алгоритми для динамічного створення тренувальних циклів, що базуватимуться на конкретних параметрах користувача.
7. Впровадити модуль дієтичного планування, здатний вираховувати добові потреби в макронутрієнтах, воді та калоріях.
8. Реалізувати інструментарій комп'ютерного зору для розпізнавання харчової цінності страв за їхніми фотографіями.
9. Створити панель моніторингу, яка дозволятиме відстежувати дотримання графіків та аналізувати загальний прогрес.
10. Закласти архітектурний фундамент для майбутнього масштабування системи, зокрема для розробки нативних мобільних застосунків.

Об'єктом дослідження виступає безпосередньо процес використання програмних алгоритмів для автоматизованої генерації індивідуальних стратегій харчування та тренувань, а також відстеження змін у фізичній формі користувачів. Предметом дослідження є технологічні та інструментальні рішення, необхідні для створення такої системи. Сюди входять архітектурні патерни, механізми інтеграції AI-моделей, сучасний веб-інструментарій та підходи до проєктування ергономічних користувацьких інтерфейсів.

Наукова інноваційність проєкту визначається створенням комплексного екосистемного підходу. Робота вперше поєднує в одному україномовному продукті генерацію спортивних програм, візуальний ШІ-аналіз харчування за фото та гнучкі інструменти моніторингу.

Практична значущість розробки полягає у випуску повністю робочого програмного рішення. Воно може використовуватися як незалежний вебсервіс для персонального супроводу клієнтів, а також слугуватиме міцною технологічною базою для подальшої монетизації та створення кросплатформного мобільного продукту.

Висновок до розділу 1

У межах першого розділу даного кваліфікаційного проєкту було виконано глибокий аналіз досліджуваної сфери, проаналізовано наявні на ринку цифрові продукти, а також визначено ключові орієнтири та конкретні завдання для наступних етапів розробки.

Нинішні умови життя, які характеризуються гіподинамією, неякісним раціоном та браком рухливості, стали причиною різкого падіння показників здоров'я в суспільстві. Зростання рівня ожиріння, поширення хвороб серця і судин, а також загальне погіршення фізичної форми перетворилися на виклики світового масштабу. З огляду на це, проєктування розумних програмних рішень, що гарантують глибоко індивідуалізований супровід у процесі фізичного вдосконалення, є вкрай нагальною потребою.

Дослідження сфери показало: для досягнення дієвих результатів необхідно паралельно вирішувати дві невіддільні задачі — грамотно вибудовувати графік навантажень та налагоджувати збалансовану дієту. Урахування цілого спектра унікальних характеристик людини (віку, статі, фізичного бекграунду, наявного інвентарю, розпорядку дня, смаків та кінцевих намірів) дає змогу багаторазово збільшити результативність тренувань і забезпечити довгостроковий ефект.

Вивчення популярних застосунків-аналогів (серед яких MyFitnessPal, Hevy, Noom, Nike Training Club, Future тощо) дозволило виявити їхні слабкі сторони. Переважна частина цих продуктів має вузький фокус, використовує стандартизовані схеми, грішить слабкою персоналізацією та практично ігнорує можливості ШІ у сфері розпізнавання їжі через комп'ютерний зір. До того ж левова частка відомих сервісів не має повноцінної української локалізації і не здатна запропонувати формат єдиного екосистемного рішення, де б органічно поєднувалися спорт, дієтологія та аналітика прогресу.

Спираючись на ці висновки, було чітко сформульовано головну мету поточного проєкту: створення розумної експертної платформи для автоматизованого генерування унікальних тренувальних і харчових циклів із паралельним відстеженням фізичних досягнень клієнта.

Заради реалізації цього задуму було виокремлено перелік конкретних кроків, а також окреслено предмет та об'єкт наукової роботи. Інноваційність дослідження ґрунтується на концепції єдиного україномовного вебпродукту, що успішно синтезує алгоритмічне конструювання фітнес-програм, обчислення персонального макросу та нейромережевий аналіз фотографій раціону.

Матеріали, напрацьовані на цьому етапі, формують надійний теоретико-практичний фундамент для наступних стадій проєкту: проєктування бази даних, створення архітектурних шаблонів, розробки математичних алгоритмів та безпосереднього написання коду для серверної та клієнтської частин системи.

Зрештою, сформовані підсумки беззаперечно доводять своєчасність і затребуваність обраного напрямку, підкреслюючи критичну необхідність впровадження високотехнологічного інтелектуального інструменту на перетині персонального фітнесу та нутриціології.

Розділ 2. Інформаційне та математичне забезпечення

2.1. Аналіз предметної області та вимоги до системи

Вивчення предметного середовища виступає фундаментальним кроком у процесі проєктування будь-якого програмного забезпечення. Завдяки цьому етапу розробник отримує змогу детально дослідити процеси, що потребують автоматизації, ідентифікувати головних стейкхолдерів, налагодити рух інформаційних потоків і висунути точні критерії до майбутнього застосунку.

Сфера застосування платформи, що проєктується, зосереджена навколо питань індивідуального тілесного вдосконалення людини. Цей напрям органічно поєднує дві глобальні, але тісно споріднені галузі: безпосередньо фітнес (регулярну рухову активність) та нутриціологію (збалансований раціон). Ключове завдання продукту полягає у забезпеченні клієнта універсальним інструментарієм, який сприятиме максимально швидкому та безпечному досягненню бажаних результатів.

Аби згенерувати дійсно індивідуальну програму, застосунок має акумулювати та опрацьовувати значний масив первинних відомостей. До переліку вхідних параметрів системи належать:

- базові антропометричні показники клієнта (тобто стать, зріст, вік і поточна маса тіла);
- оцінка поточного фізичного стану, спортивний досвід та інформація про перенесені пошкодження;
- головний вектор роботи (бажання схуднути, наростити м'язову масу, покращити витривалість чи просто підтримати здоров'я);
- характер повсякденної активності та стиль життя;
- наявність спортивного інвентарю для занять (умови власної оселі, повноцінний спортзал або частковий набір тренажерів);
- гастрономічні вподобання, можливі алергічні реакції та специфічні дієтичні ліміти.

Відштовхуючись від зібраної інформації, платформа має видавати комплексний згенерований результат. Вихідні дані програми охоплюють формування:

- унікальної програми тренувань, розрахованої на певний часовий проміжок (тижні чи місяці), із чіткою регламентацією кожної вправи, кількості підходів, повторень та інтервалів відпочинку;
- персональної схеми харчування, що базується на вирахуваній денній нормі калорій та балансі білків, жирів і вуглеводів;
- порад щодо дотримання питного режиму та вживання додаткових нутрієнтів;
- результатів візуального розпізнавання сфотографованих страв із визначенням того, наскільки цей раціон відповідає кінцевій меті;
- аналітичних звітів про динаміку розвитку користувача разом із рекомендаціями щодо необхідних змін у плані.

Функціональні критерії окреслюють ключові можливості продукту. Додаток повинен мати надійну систему реєстрації та входу, зокрема із застосуванням облікового запису Google. Після детального заповнення особистої анкети користувачем, платформа має автоматизовано конструювати персональні графіки спортивних навантажень та харчування. Вагомою складовою функціоналу є здатність обробляти знімки їжі, визначаючи її харчову цінність і ступінь відповідності запланованим макросам. Окрім цього, програма зобов'язана фіксувати історію харчування і тренувань, моніторити досягнення та пропонувати подальші кроки.

Нефункціональні вимоги мають не меншу вагу у процесі проєктування. До цієї групи належать: миттєве генерування контенту, ергономічний та сучасний дизайн користувацького інтерфейсу, кросплатформна адаптивність, гарантування безпеки приватних даних, стабільність функціонування та потенціал до масштабування архітектури в перспективі. Додатково вимагається забезпечення мультимовності, що включає українську, англійську та польську локалізації.

Отже, ретельне дослідження предметного середовища дало змогу однозначно ідентифікувати базові бізнес-процеси, напрямки потоків інформації, а також

функціональні й технічні норми. Здобуті висновки формують міцний базис для подальшого проєктування системної архітектури, створення моделі бази даних та безпосереднього написання коду застосунку.

2.2. Проектування бази даних

Розробка оптимальної структури зберігання інформації є одним із найбільш відповідальних кроків під час створення інтелектуальної експертної системи, оскільки її архітектурна якість безпосередньо визначає показники швидкодії, потенціал до масштабування, відмовостійкість інформаційного сховища, а також загальну ефективність подальших етапів інженерії програмного забезпечення.

Для проєктуваного програмного комплексу було обрано реляційну концепцію організації даних, реалізовану за допомогою системи управління базами даних PostgreSQL. Такий вибір аргументований необхідністю підтримки складних реляційних зв’язків між об'єктами, забезпеченням транзакційної цілісності, наявністю вбудованих інструментів контролю даних, а також нативною сумісністю та високою зручністю інтеграції з ORM-фреймворком Entity Framework Core.

Логічна модель сховища спроєктована відповідно до критеріїв третьої нормальної форми (3NF), проте з обґрунтованим застосуванням елементів денормалізації в окремих структурах задля максимізації швидкості виконання операцій читання. Архітектура бази даних об'єднує понад 15 ключових таблиць, які за функціональною ознакою можна розподілити на такі логічні блоки: профілі та автентифікація користувачів, системні довідники, модулі тренувальних програм, підсистеми обліку харчування й моніторингу стану, а також компоненти адміністрування та підписок.

До базових сутностей системи належать таблиці Users та UserProfiles. Перша з них призначена для збереження первинних автентифікаційних відомостей, включаючи унікальний ідентифікатор, адресу електронної пошти, ім’я, системну роль та часову мітку створення облікового запису. У таблиці UserProfiles акумулюються детальні біометричні та фізіологічні показники людини: її стать, дата народження, лінійні та вагові параметри (зріст, вага), посилання на тип щоденної

активності (LifestyleId), а також наявні дієтичні обмеження та специфічні алергічні реакції. Подібний розподіл інформації дає змогу ізолювати процеси автентифікації від обробки чутливих персональних даних, підвищуючи гнучкість управління системою.

Нормалізацію даних та ефективне управління контентом забезпечує набір довідникових таблиць, серед яких MuscleGroups, Equipments, Exercises, Goals, Lifestyles, Statuses та SubscriptionPlans. Ключова роль у цьому блоці належить таблиці Exercises, що містить детальні відомості про кожну фізичну вправу: її найменування, текстовий опис техніки виконання, URL-посилання на відеодемонстрацію, градацію складності, а також зовнішні ключі для встановлення реляційних зв'язків із відповідними групами м'язів та необхідним спортивним інвентарем.

Однією з фундаментальних підсистем сховища є блок організації спортивних програм. Таблиця WorkoutPlans акумулює узагальнені параметри плану, зокрема прив'язку до користувача, обрану мету, тривалість циклу, рівень навантаження та поточний статус. Дана сутність пов'язана відношенням із таблицею PlanDays, яка структурує окремі тренувальні дні. Стосовно таблиці PlanDays, вона має зв'язок із проміжною структурою PlanExercises, що деталізує конкретні рухи в межах дня, фіксуючи кількість робочих підходів, повторень, тривалість інтервалів відпочинку у секундах, а також текстові інструкції та нотатки, згенеровані штучним інтелектом.

Дієтологічний контент організовано за допомогою таблиць DailyNutritionGoals та MealLogs. Кожен запис про добові цілі харчування (DailyNutritionGoals) інтегрується із конкретним днем спортивного плану та регламентує цільовий калораж, а також необхідні обсяги споживання білків, жирів, вуглеводів і води. У таблиці MealLogs ведеться щоденна фіксація кожного прийому їжі, де зберігаються посилання на завантажені фотознімки страв, розпізнані макронутрієнти, автоматична оцінка відповідності встановленим нормам та інтелектуальний відгук системи. Для супроводу рекомендаційних алгоритмів ШІ впроваджено таблицю NutritionRecommendations, яка акумулює унікальні добові поради щодо раціону. Крім того, архітектура передбачає таблиці для адміністрування системних ролей (Roles),

станів планів (Statuses), користувацьких підписок (UserSubscriptions) та фіксації історії змін (AuditLogs у довгостроковій перспективі).

Взаємозв’язки між таблицями

Проєкт бази даних спирається на розгалужену систему реляційних зв'язків. Ключовими типами відношень є зв'язки «один-до-багатьох» (one-to-many), що зв'язують, наприклад, одного користувача з його планами тренувань або один план із його днями, а також «багато-до-багатьох» (many-to-many), що застосовуються для відображення належності вправ до різних груп м'язів чи наявності кількох підписок у користувача. Для побудови відношень багатьох до багатьох розроблено спеціалізовані проміжні таблиці.

Особлива увага приділена підтриманню референтної цілісності даних. Усі зовнішні ключі забезпечені правилами каскадного видалення або відповідними обмеженнями залежно від логіки бізнес-процесів. Задля збереження критично важливої інформації (профілі, тренувальні плани, логи дієти) реалізовано концепцію м'якого видалення (Soft Delete) за допомогою атрибутів IsDeleted та DeletedAt.

Оптимізація та продуктивність

З метою забезпечення високої швидкодії системи для полів, які найчастіше беруть участь у операціях пошуку та фільтрації (таких як UserId, WorkoutPlanId, DailyNutritionGoalId, CreatedAt), було спроєктовано та створено індекси. Додатково впроваджено композитні індекси для прискорення складних комплексних запитів, наприклад, для швидкої вибірки всіх днів конкретної програми або отримання історії логів харчування за обраний календарний період.

Процес зберігання фотографій страв реалізовано шляхом фіксації текстового шляху до файлу в базі даних, тоді як самі медіафайли розміщуються у зовнішньому об'єктному хмарному сховищі Cloudflare. Такий підхід дає змогу уникнути надмірного збільшення обсягу бази даних. Побудована ER-діаграма унаочнює структуру бази даних, відображаючи взаємозв'язки між основними сутностями та таблицями.

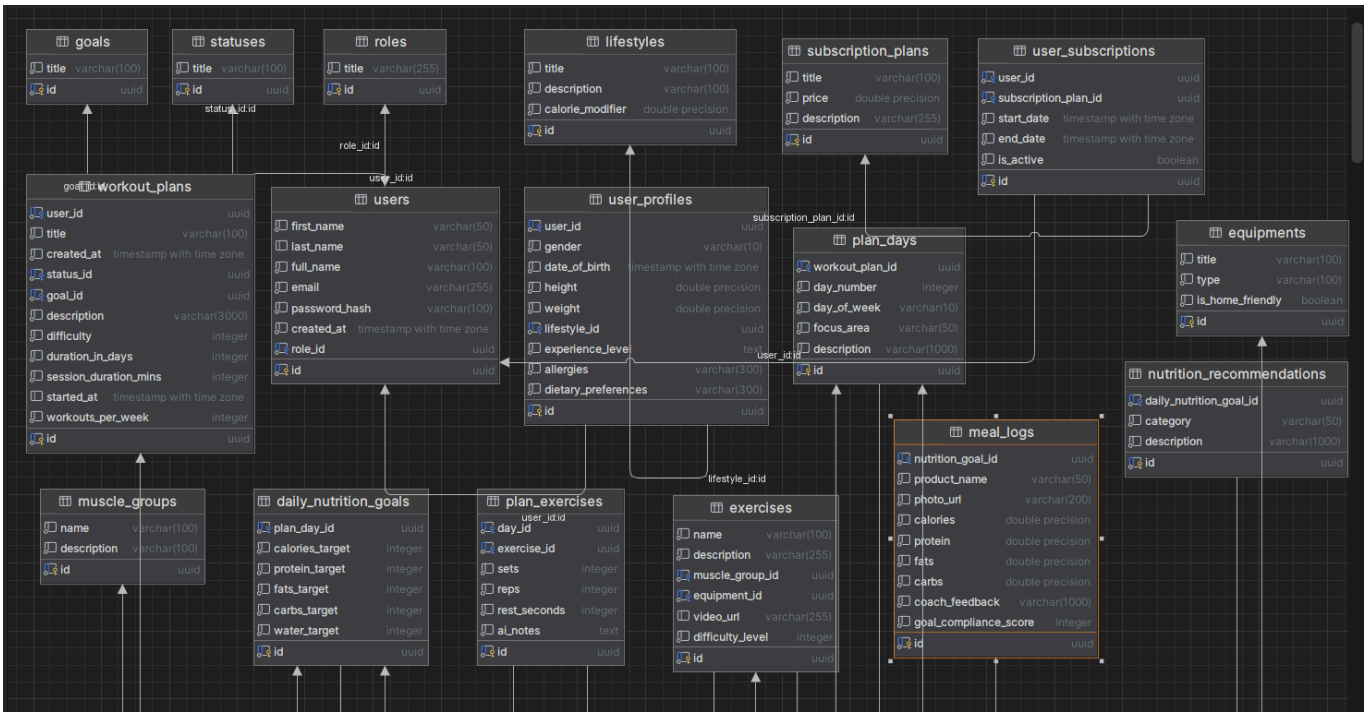

Рис. 2.1. ER-діаграми бази даних

Джерело: розроблено автором

Безпека даних

Під час проєктування сховища даних було враховано жорсткі вимоги щодо захисту персональної інформації. У системі функціонує розмежування прав доступу на основі рольової моделі, забезпечено криптографічне шифрування конфіденційних токенів, а також налаштовано механізми логування аудиту ключих операцій, що здійснюються користувачами та представниками адміністративного персоналу.

У підсумку, створена модель бази даних повністю відповідає всім функціональним вимогам до системи, гарантує високу швидкість обробки запитів, масштабованість структури та захищеність збережених відомостей, що є надійним технологічним фундаментом для розробки як серверної (Backend), так і клієнтської (Frontend) частин інтелектуальної експертної системи Vector.

2.3. Архітектура системи

Побудова програмної архітектури виступає наріжним каменем для успішного функціонування будь-якого розгалуженого цифрового комплексу. Під час проєктування інтелектуальної експертної платформи Vector ключовий акцент було зроблено на застосуванні прогресивної, адаптивної та масштабованої структури, що

цілковито корелює з актуальними індустріальними стандартами програмної інженерії періоду 2025–2026 років.

Як базовий архітектурний каркас було затверджено Onion Architecture (чисту архітектуру), відому в науковій спільноті також як Clean Architecture чи патерн Ports and Adapters. Зазначену концепцію було розширено за допомогою патерну розділення відповідальності команд і запитів CQRS (Command Query Responsibility Segregation) та успішно реалізовано на практиці завдяки інструментарію бібліотеки MediatR. Такий інтегрований підхід забезпечує глибоку ізоляцію компонентів, технологічну незалежність ядра від зовнішніх сервісів, а також створює чудові умови для проведення всебічного тестування системи.

Концепція Onion Architecture базується на розшаруванні системи у вигляді концентричних кіл, де кожен рівень має чітко окреслені межі. Ядром усього комплексу є доменний шар (Domain Layer), який акумулює в собі найважливіші елементи: чисті бізнес-правила, сутності предметної області, об'єкти-значення (value objects) та внутрішні доменні події. Цей внутрішній рівень є абсолютно автономним і не залежить від баз даних, користувацького інтерфейсу, сторонніх фреймворків чи будь-яких зовнішніх технологічних рішень. Завдяки такій повній ізоляції логіка бізнес-процесів залишається стабільною та захищеною від змін у зовнішньому технологічному стеку.

Над доменним ядром розташовується прикладний шар (Application Layer), головне завдання якого полягає в координації та управлінні бізнес-сценаріями. У межах цього рівня кожна користувацька дія (Use Case) інкапсулюється у відповідну структуру запиту (Queries) або команди (Commands). Обробка цих командних та інформаційних об'єктів здійснюється через поведінковий патерн «Медіатор», впроваджений за допомогою бібліотеки MediatR. Застосування медіатора дозволяє суттєво розвантажити програмний код контролерів, роблячи їх максимально лаконічними та ізольованими. Окрім того, прикладний шар відповідає за первинну перевірку вхідних параметрів через FluentValidation, трансформацію об'єктів за допомогою AutoMapper та збереження загальної логіки, яка безпосередньо не належить до сутностей домену.

Технічну реалізацію та взаємодію із зовнішнім середовищем забезпечує інфраструктурний шар (Infrastructure Layer). Тут зосереджено конкретні механізми доступу до даних на базі Entity Framework Core, сервіси керування сховищами медіаконтенту (відеоінструкцій до вправ та фотографій раціону), інтеграційні модулі для зв'язку зі штучним інтелектом, налаштування інструментів ідентифікації та авторизації (включаючи токени JWT та протокол Google OAuth 2.0), а також компоненти логування й кешування запитів. Даний рівень функціонально підпорядкований прикладному та доменному шарам, що повністю відповідає ключовому правилу залежностей Чистої архітектури.

Рівень представлення (Presentation або API Layer) спроєктовано як вебсервіс на базі технології ASP.NET Core Web API. Його головна місія — перехоплення HTTP-запитів від клієнтського застосунку, передача їх для подальшого опрацювання в MediatR та повернення згенерованих відповідей назад на фронтенд. Згідно з парадигмою Thin Controllers, API-контролери повністю позбавлені прямої логіки обробки даних і виконують виключно маршрутизаційну роль.

Важливою інноваційною рисою архітектурного каркаса є інтеграція підходу CQRS. Операції, пов'язані з вибіркою та читанням інформації (наприклад, перегляд списку вправ, згенерованих програм або історії дієти), інкапсульовані в межах запитів (Queries), тоді як транзакційні дії зі зміни стану бази (створення нових планів, фіксація логів їжі, оновлення анкети профілю) виконуються суворо через команди (Commands). Таке логічне розмежування відкриває можливості для незалежної оптимізації кожної гілки: під час читання застосовуються проекції, оптимальні індекси та метод `.AsNoTracking()`, а під час виконання команд — повноцінні транзакції та механізми Change Tracking.

Задля безпечного обміну даними між різними компонентами системи було розроблено та впроваджено політику CORS (Cross-Origin Resource Sharing). На рівні сервера чітко регламентовано дозволені джерела запитів (origins), підтримувані HTTP-методи (GET, POST, PUT, DELETE), заголовки та правила передачі облікових даних (Credentials). Це гарантує захищену та стабільну взаємодію з React-клієнтом, навіть якщо він розгорнутий на іншому мережевому порту чи сторонньому домені.

Клієнтська компонента (Frontend) також побудована відповідно до сучасних інженерних концепцій на базі бібліотеки React 18 із використанням строгої типізації TypeScript. Керування глобальним станом інтерфейсу та ефективне кешування серверних відповідей покладено на Redux Toolkit у поєднанні з технологією RTK Query. Структурування вихідного коду реалізовано за методологією Feature-Sliced Design (FSD), де кожен функціональний блок (такий як Workout, Nutrition, Auth, Profile або Admin) інкапсулює всередині себе власні компоненти, зрізи стану (slices), сервісні модулі та типи даних. Такий модульний розподіл суттєво полегшує процеси довгострокового супроводу великого проєкту та оптимізує паралельну командну розробку.

Візуальне оформлення та стилізація елементів виконані за допомогою утилітарного фреймворку Tailwind CSS та бібліотеки готових компонентів Material Tailwind. Це дозволило створити прогресивний, лаконічний та швидкий користувацький інтерфейс, адаптований під фірмову темну тему (dark theme). Комунікація між клієнтом і сервером базується на принципах архітектурного стилю REST API. Передача великих файлів (зокрема, відеоматеріалів або фотографій страв) організована через запити типу `multipart/form-data` із візуалізацією індикаторів завантаження (progress bars) на стороні інтерфейсу. Процес автентифікації спирається на використання захищених JWT-токенів, що зберігаються в локальному сховищі браузера (localStorage) та автоматично оновлюються в міру закінчення терміну дії.

Спроєктована архітектурна модель заздалегідь орієнтована на майбутнє масштабування та модернізацію системи. Завдяки чіткому розмежуванню обов'язків між шарами у довгостроковій перспективі стає можливим:

- делегування процесів генерації тренувальних планів в ізольований незалежний мікросервіс;
- впровадження технології GraphQL для гнучкішого формування клієнтських запитів до даних;
- інтеграція високопродуктивного сховища Redis для кешування інформації у пам'яті;

- перехід до Event-Driven архітектури за допомогою інструментів MediatR Notifications.

Задля централізованої обробки непередбачуваних збоїв на загальносистемному рівні розроблено спеціалізований проміжний шар Exception Middleware, який перехоплює всі виняткові ситуації, забезпечує їх логування та повертає на клієнтську сторону стандартизовані й зрозумілі відповіді з коректними HTTP-статусами.

Таким чином, архітектурний каркас системи Vector є прогресивним, адаптивним, придатним до тестування та масштабованим. Він повністю відповідає рівню складності поставленого інженерного завдання — побудові інтелектуального сервісу з глибокою бізнес-логікою, бездоганною ШІ-інтеграцією та високими критеріями до якості взаємодії з користувачем. Практичне впровадження Onion Architecture та патерну CQRS гарантує можливість ефективного супроводу, рефакторингу та довгострокового розвитку програмного продукту.

2.4. Математичне та алгоритмічне забезпечення

Фундаментом інтелектуальної експертної системи Vector виступає її математично-алгоритмічна база, адже вона відповідає за безпомилковість обчислень, глибину індивідуалізації та релевантність згенерованих порад. Від того, наскільки коректно спроєктовані математичні моделі, безпосередньо залежить результативність тренінгових схем, вірогідність дієтологічних вказівок та, як наслідок, загальна корисність платформи для кінцевого споживача.

Ключовими показниками для розрахунку енергетичних потреб організму є визначення базового метаболізму (BMR) спільно із сукупною добовою витратою енергії (TDEE). Щоб обчислити показник BMR, платформа застосовує алгоритм Міффліна-Сан Жеора, який визнано одним із найбільш релевантних та точних для дорослих осіб:

- Зокрема, для чоловічої статі формула має такий вигляд : $BMR = 10 \times \text{вага(кг)} + 6{,}25 \times \text{зріст(см)} - 5 \times \text{вік(років)} + 5$.

- Відповідно, для жіночої статі розрахунок ведеться так : BMR = 10 × вага(кг) + 6,25 × зріст(см) − 5 × вік(років) − 161.

Знайдений індекс BMR підлягає подальшому коригуванню через коефіцієнт фізичної активності (PAL), що прямо залежить від вказаного в анкеті стилю життя (Lifestyle). Програмний комплекс виокремлює чотири градації щоденної активності: малорухливий або Sedentary (1.2), злегка активний або Lightly Active (1.375), помірно активний або Moderately Active (1.55) та високоактивний або Very Active (1.725). Завдяки цим множникам система формує остаточну індивідуальну денну норму кілокалорій (TDEE).

Наступним кроком є адаптація розрахованої калорійності під конкретну мету користувача (Goal):

- Для зниження ваги передбачено створення енергетичного дефіциту в межах 15–25 % від показника TDEE.
- Для нарощування м'язових об'ємів формується профіцит, що становить 10–20 % від TDEE.
- Для рекомпозиції тіла (паралельного спалювання жиру та росту м'язів) калораж залишається на рівні підтримки або з мінімальним відхиленням у той чи інший бік.
- Для збереження поточної фізичної форми споживання калорій дорівнює обчисленому TDEE.

Після затвердження добового калоражу відбувається калькуляція макронутрієнтів (БЖВК), яка базується на динамічних коефіцієнтах, що змінюються відповідно до статі, спортивного стажу та поставленої мети. Стандартний розподіл нутрієнтів має наступні пропорції:

- Обсяг білків варіюється від 1,8 до 2,6 г на кожен кілограм ваги, причому верхня межа застосовується за умови інтенсивного тренінгу для росту м'язів.
- Кількість жирів фіксується на рівні, не меншому за 0,8–1,0 г на кілограм маси тіла, що критично важливо для збереження здорового гормонального фону.
- Вуглеводи покривають усю залишкову кількість калорій після вирахування норм білків та жирів.

Також під час калькуляції алгоритми беруть до уваги специфічні обмеження: наявність алергічних реакцій, тип дієти (кето, вегетаріанство тощо) та асортимент продуктів, які вже є в меню клієнта.

Що стосується модуля генерування спортивних планів, він виступає одним із найскладніших технічних елементів платформи і функціонує за мультифакторним принципом, враховуючи:

- головну ціль людини;
- поточну спортивну кваліфікацію (від початківця до профі);
- наявність тренувального інвентарю (від повноцінного залу до виключно домашніх умов);
- пріоритетні для розвитку м'язи або ті, що потребують виключення з програми;
- бажану тривалість одного заняття та їх кількість на тиждень;
- наявність фізичних ушкоджень у перспективі розвитку системи.

Процес побудови починається з формування каркаса плану (розподіл днів, м'язовий фокус), після чого алгоритм заповнює його вправами з довідника за ієрархічним принципом: на початку сесії ставляться важкі багатосуглобові рухи, а наприкінці — ізолювальні. Кожній вправі присвоюється оптимальна кількість повторень та підходів, що диктується метою: для розвитку сили це 3–6 повторень, для збільшення м'язів — 8–12, а для тренування витривалості — від 12 до 20.

Алгоритмічна база дієтологічного модуля функціонує у тісній синергії з тренувальним розкладом. Вона аналізує поточні залишки енергетичної квоти та нутрієнтів до кінця доби, формуючи відповідні пропозиції страв. Вирішенням цього завдання займається спеціальний інструмент Meal Synthesizer, який вибудовує персональні рекомендації, відштовхуючись від нерозподілених макросів та доступної продуктової бази.

Унікальним аспектом математичної архітектури є застосування ШІ, що реалізовано у двох ключових векторах:

1. Технології комп'ютерного зору, призначені для візуального аналізу їжі. Отримавши фотознімок, інтелектуальна модель розпізнає страву, оцінює її орієнтовну масу (або бере дані, введені юзером) та з високою точністю

вираховує її хімічний склад. Здобуті показники зіставляються із денними нормативами, генеруючи загальний бал відповідності (Goal Compliance Score).

2. Інструментарій генеративного штучного інтелекту, за допомогою якого створюються розгорнуті описи страв, їхні назви, дієтичні підказки та зворотний зв'язок. Для цього залучаються передові методики промпт-інжинірингу, що адаптують результати видачі під мову клієнта, його вподобання та кінцеву мету.

Абсолютно всі обчислювальні механізми спроєктовані з фокусом на безпеку: система автоматично перевіряє результати на відповідність фізіологічним нормам (наприклад, блокує критичне зниження білка), залишаючи при цьому можливість ручного редагування показників тренером або самим юзером.

Підсумовуючи, алгоритмічна та математична складова платформи Vector є вдалим симбіозом перевірених часом дієтологічних формул, інноваційних підходів до конструювання планів та потужностей нейромереж. Такий комплексний підхід дає змогу продукувати глибоко індивідуальні, високоякісні рекомендації, що зважають на безліч особистих чинників і вміють гнучко адаптуватися під динаміку фізіологічних змін організму.

Висновок до розділу 2

Підсумовуючи результати, отримані під час виконання другого розділу кваліфікаційної роботи, слід зазначити, що було успішно сформовано базовий інформаційний, архітектурний та математичний фундамент для інтелектуальної експертної платформи Vector. Реалізований етап системного проєктування відіграє визначальну роль у структурі дослідження, оскільки він забезпечує логічний місток від теоретичного вивчення предметного середовища до безпосередньої програмної реалізації та розробки вихідного коду продукту.

Завдяки всебічному дослідженцю специфіки предметної області вдалося детально описати та формалізувати технічні вимоги (як функціонального, так і нефункционального характеру) до розроблюваного застосунку. Зокрема, було встановлено, що досягнення високої точності індивідуалізації вимагає від системи здатності оперувати значним обсягом первинних відомостей про користувача,

включаючи його антропометрію, спортивні цілі, поточний рівень підготовки, наявний реманент та дієтичні ліміти. Водночас архітектура системи повинна спиратися на розгалужену бізнес-логіку, яка синхронізує між собою тренувальний процес та щоденне меню. Особливе місце серед технічних критеріїв посідають параметри горизонтального й вертикального масштабування, підтримка мультиязичності інтерфейсу, а також організація захищеного зв'язку із зовнішніми сервісами штучного інтелекту.

Вдалим інженерним кроком стало моделювання стійкої та оптимізованої схеми реляційного сховища даних. Спроєктована концептуальна ER-діаграма наочно ілюструє розгалужену мережу реляційних зв'язків між центральними компонентами системи, зокрема: анкетними профілями користувачів, каталогами вправ (із деталізацією по обладнанню та м'язових групах), алгоритмічно створеними фітнес-програмами (структурованими до рівня тренувальних днів і конкретних підходів), а також денними лімітами макронутрієнтів та щоденниками харчування. Затверджена нормалізована архітектура даних гарантує високу референційну цілісність, ліквідує дублювання інформації та створює сприятливі умови для оперативного виконання транзакцій, що є критично важливим для підтримки високої швидкодії вебплатформи.

Ретельне проєктування системної архітектури дозволило закласти гнучкий та стійкий до навантажень каркас для серверної компоненти проєкту. У роботі логічно обґрунтовано доцільність інтеграції концепції предметно-орієнтованого проєктування (Domain-Driven Design) із шаблонами чистої архітектури (Clean Architecture). Завдяки імплементації патерну CQRS вдалося на логічному рівні чітко розмежувати потоки для зчитування інформації та виконання транзакційних дій зі зміни стану. Зазначене архітектурне підхід істотно полегшує процеси ізольованого масштабування навантажених підсистем (таких як ресурсомісткі модулі розпізнавання фотознімків страв або автоматичний розрахунок тренувань), підвищує читабельність і придатність вихідного коду до довгострокового супроводу, а також повністю відокремлює доменне ядро від інфраструктурних деталей та зовнішніх інтерфейсів.

Вагоме наукове та практичне значення в межах даної частини дослідження має сформоване математичне й алгоритмічне забезпечення. Було чітко структуровано математичні моделі для точного вирахування базового обміну речовин, сукупної добової витрати енергії, а також індивідуального балансу макронутрієнтів (білків, жирів, вуглеводів) відповідно до визначеного користувачем вектора розвитку (схуднення, гіпертрофія м'язів чи рекомпозиція тіла). Поряд із математичними рівняннями, було спроєктовано логічні алгоритми конструювання спортивних циклів, що спираються на принципи періодизації навантажень, враховують швидкість відновлення організму та динамічно підлаштовуються під наявний інвентар.

На додаток, формалізовано інноваційні підходи щодо інтеграції штучного інтелекту в архітектуру системи Vector. Детально розписано принципи побудови спеціалізованих контекстних промптів для AI-моделей, завдяки чому досягається динамічне конструювання унікальних фітнес-програм та повністю нівелюється використання фіксованих шаблонів. Також успішно запрограмовано механізм дієтичного аналізу, що за допомогою комп'ютерного зору трансформує графічні дані із фотографій страв у структуровані цифрові відомості про нутрієнтний склад, максимально автоматизуючи контроль раціону та забезпечуючи користувача якісним зворотним зв'язком.

Таким чином, напрацювання другого розділу дозволили в повному обсязі розв'язати інженерну задачу проєктування внутрішньої логіки, архітектурних зв'язків та інфраструктури застосунку. Побудовані математичні моделі, розроблена схема бази даних та обрані архітектурні патерни формують стійку та надійну технологічну платформу для переходу до наступного етапу — безпосередньої програмної реалізації серверної (Backend) та клієнтської (Frontend) частин інтелектуальної системи Vector.

Розділ 3. Програмне та технічне забезпечення

3.1. Засоби розробки та технологічний стек

Визначення оптимального набору технологій є одним із найвідповідальніших кроків під час проєктування будь-якої програмної платформи. Саме від цього вибору безпосередньо залежать темпи програмування, а також такі критичні характеристики, як потенціал до подальшого розширення, загальна швидкодія, рівень захищеності та зручність технічного супроводу. Задля реалізації розумної системи Vector було сформовано передовий технологічний стек, який стовідсотково корелює зі світовими інженерними практиками (Best Practices) та створює ідеальні умови для впровадження глибокої бізнес-логіки на основі ШІ.

Серверний сегмент (Backend) Фундаментом для серверної інфраструктури було обрано новітню платформу .NET 10, що працює в тандемі з об'єктно-орієнтованою мовою C#. Цей технологічний каркас давно визнаний індустрією як один із найпотужніших засобів для створення стійких до великих навантажень серверних рішень. Обмін даними налаштовано за допомогою архітектурного стилю Web API.

Щоб гарантувати високу модульність та дотриматися парадигм чистої архітектури (Clean Architecture) і DDD (Domain-Driven Design), до проєкту було підключено інструмент MediatR. Завдяки цій бібліотеці вдалося успішно імплементувати шаблон CQRS, що дозволило логічно відокремити потоки виконання інформаційних запитів (Queries) від команд, які змінюють стан системи (Commands). Це рішення радикально зменшує взаємозалежність компонентів і значно полегшує написання тестів.

Процес перевірки коректності вхідних параметрів покладено на бібліотеку FluentValidation, що дає змогу вилучити громіздкі алгоритми валідації з доменного ядра, залишаючи код чистим. Документування інтерфейсів програмування додатків (API) та забезпечення зручного середовища для їх тестування реалізовано за допомогою сервісу Scalar API docs.

Для комунікації зі сховищем даних інтегровано передову систему об'єктно-реляційного відображення (ORM) — Entity Framework Core. Роль головної системи управління реляційними базами (СУБД) виконує PostgreSQL. Аби уникнути конфліктів між різними середовищами розробки та забезпечити стабільність, базу даних ізольовано шляхом розгортання у контейнерах Docker. Додатковий рівень абстракції між бізнес-логікою та механізмами зчитування даних забезпечується завдяки патерну Repository.

Клієнтський інтерфейс (Frontend) Візуальну та інтерактивну частину застосунку побудовано навколо екосистеми React 18, що застосовується у зв'язці зі строго типізованим інструментарієм TypeScript. Наявність статичної типізації відіграє вирішальну роль у масштабних проєктах, адже вона дозволяє перехоплювати помилки ще на стадії збірки та значно полегшує роботу з розгалуженими об'єктами (як-от структурами фітнес-програм чи щоденниками харчування).

За швидку збірку модулів відповідає бандлер Vite, який не лише оптимізує підготовку коду до продакшену, але й гарантує миттєве оновлення компонентів (HMR) у процесі програмування. Робота з глобальним станом інтерфейсу (State Management) довірена технології Redux та її осучасненому набору утиліт Redux Toolkit (RTK). Переходи між сторінками без перезавантаження браузера (навігація у форматі SPA) організовано через React Router.

Візуальний дизайн платформи, виконаний у сучасній темній стилістиці (dark theme), спроєктовано завдяки utility-first фреймворку Tailwind CSS. Він дає змогу оперативно розробляти кросбраузерні та адаптивні компоненти, звільняючи від потреби писати та підтримувати важкі файли стилів. Оскільки проєкт має міжнародний потенціал, систему обладнано інструментом i18next, що підтримує переклад інтерфейсу українською, англійською та польською мовами.

Інтеграції, захист та інфраструктурні рішення Процеси автентифікації та захисту персональних відомостей базуються на передових стандартах: вхід реалізовано через провайдера Google OAuth, а захист маршрутів API гарантується криптографічними токенами JWT (JSON Web Tokens).

Фундаментальною складовою продукту є тісна взаємодія зі штучним інтелектом. За виконання таких складних завдань, як алгоритмічна побудова спортивних планів, візуальна оцінка сфотографованих страв (Meal Photo Analysis), розрахунок БЖВК та генерація унікальних рецептів, відповідають API передових нейромереж (зокрема моделей OpenAI або Claude). Важливо підкреслити, що вся комунікація з ШІ відбувається ізольовано на бекенді, що гарантує надійне збереження API-ключів та оптимізує керування запитами (промптами).

Зрештою, розгортання робочої версії платформи здійснено на базі потужної хмарної інфраструктури Cloudflare. Використання хмарних технологій забезпечує максимальну доступність вебсервісу, надійний захист інформації користувачів та відкриває шляхи для швидкого масштабування серверних потужностей у моменти пікового навантаження системи Vector.

3.2. Вимоги до технічного та програмного забезпечення

Встановлення чітких апаратних та програмних критеріїв виступає фундаментальним кроком під час проєктування будь-якої новітньої інформаційної інфраструктури. Зважаючи на те, що сервіс Vector функціонує за класичною клієнт-серверною моделлю та залучає ресурсомісткі інструменти обробки інформації (зокрема технології комп'ютерного зору та взаємодію з нейромережами), виникає гостра необхідність у детальному розмежуванні параметрів для сервера, клієнтських гаджетів та робочого місця інженера. Адекватно підібрані технічні характеристики є запорукою безперебійного функціонування, швидкої реакції інтерфейсу та раціонального розподілу обчислювальних потужностей.

Щоб гарантувати продуктивну роботу серверного боку (Backend), який розміщується на хмарних потужностях AWS, потрібно виділити належні апаратні ресурси. Оскільки опрацювання ключових бізнес-правил, генерація планів штучним інтелектом та робота бази даних PostgreSQL (у контейнерах Docker) відбуваються саме на сервері, інфраструктура вимагає солідного запасу потужності. Базові критерії для віртуального сервера (VDS/VPS) або хмарного середовища передбачають наявність мінімум двох віртуальних ядер процесора (2 vCPU) нового покоління та оперативної пам'яті обсягом від 4 до 8 гігабайтів. Обов'язковим є застосування швидкісних твердотільних SSD-накопичувачів ємністю від 30 ГБ, що гарантують миттєвий доступ до бази даних і надійне збереження медіаконтенту

(відеороликів до вправ та знімків раціону). Програмна основа сервера має спиратися на стабільні дистрибутиви ОС Linux (наприклад, Debian чи Ubuntu Server), адже вони забезпечують найвищий ступінь кіберзахисту та ідеальну сумісність із технологіями контейнеризації. Додатково на сервісному вузлі повинні бути розгорнуті середовище виконання .NET 10 Runtime та система Docker Engine.

Що стосується клієнтського інтерфейсу, його реалізовано за концепцією односторінкового застосунку (SPA), що делегує виконання частини операцій (рендеринг компонентів та управління станом через Redux) безпосередньо браузеру користувача. Втім, завдяки глибокій оптимізації збірки через інструмент Vite, системні вимоги до пристроїв кінцевих споживачів залишаються вкрай низькими та демократичними. Платформа є абсолютно кросплатформною і стабільно функціонує на смартфонах, планшетних комп'ютерах, лептопах чи стаціонарних ПК. Апаратний мінімум для користувацького гаджета обмежується процесором із тактовою частотою від 1,5 ГГц та наявністю 2 ГБ оперативної пам'яті. Критично важливою умовою для розкриття повного функціоналу платформи (а саме роботи сканера їжі Meal Photo Analysis) є наявність вбудованої цифрової камери для фотографування страв. Програмні вимоги зводяться до наявності оновленого веббраузера (Google Chrome, Safari, Microsoft Edge або Mozilla Firefox) з підтримкою актуальних стандартів ECMAScript 6, CSS3 та HTML5. З огляду на те, що застосунок постійно обмінюється даними із сервером, завантажує відеоінструкції та надсилає зображення до ШІ, безперервне інтернет-з'єднання зі швидкістю не менш як 5 Мбіт/с є обов'язковою вимогою.

Значної уваги також потребує організація робочого простору розробника, адже процеси масштабування та підтримки проєкту вимагають потужного обладнання. Створення full-stack платформи із залученням мікросервісних підходів, одночасним функціонуванням фронтенду, бекенду та локальних Docker-контейнерів для баз даних генерує суттєве навантаження на систему. Тому комп'ютер інженера повинен мати щонайменше 4-ядерний фізичний процесор, швидкий SSD-накопичувач та не менше 16 ГБ оперативної пам'яті, аби уникнути дефіциту ресурсів під час паралельної роботи середовищ розробки. Стандартний програмний інструментарій розробника має охоплювати операційну систему (Linux, macOS чи Windows), актуальну версію Node.js (LTS), набір .NET SDK 10 та клієнт Docker Desktop для розгортання локальної інфраструктури. Для написання коду найоптимальніше використовувати прогресивні IDE, такі як JetBrains Rider, Microsoft Visual Studio 2022 або гнучкий редактор Visual Studio Code. Контроль версій проєкту здійснюється виключно через систему Git.

У підсумку, чітке дотримання сформульованих апаратно-програмних критеріїв є запорукою відмовостійкості експертної системи Vector на всіх стадіях її життєвого

циклу: починаючи від локального тестування програмістами і закінчуючи розгортанням у хмарному середовищі та масовою експлуатацією клієнтами.

3.3. Розробка серверної частини (Backend)

Створення бекенду виступає найбільш масштабним і ресурсомістким кроком у конструюванні інтелектуальної платформи Vector. На цьому рівні сконцентрована ключова бізнес-логіка, механізми перевірки та обробки інформації, протоколи безпеки, а також розгалужені алгоритми комунікації з нейромережами. Від того, наскільки грамотно спроєктована архітектура серверної інфраструктури, прямо залежить загальна швидкодія застосунку, його відмовостійкість та здатність до безпроблемного масштабування.

Архітектурні парадигми та патерни Фундаментом для серверного коду слугує симбіоз чистої архітектури (Clean Architecture) та принципів предметно-орієнтованого проєктування (DDD). Завдяки такій структурі доменне ядро, яке містить найважливіші бізнес-правила та сутності (як-от UserProfile, WorkoutPlan, DailyNutritionGoal), повністю ізольоване від баз даних, зовнішніх фреймворків чи мережевих протоколів. Задля уникнення помилок із передачею ідентифікаторів та забезпечення суворої типізації було впроваджено Strongly Typed IDs (наприклад, UserId, WorkoutPlanId). Керування потоками даних та розподіл обов'язків реалізовано через патерн CQRS за допомогою інструменту MediatR. Такий підхід гарантує логічне розмежування процесів. Команди, що модифікують стан системи (Commands, як-от CreateWorkoutPlanCommand), інкапсулюються у спеціальні класи та маршрутизуються через інтерфейс ISender. Водночас запити на читання (Queries) виконуються через окремі інтерфейси, що дає змогу звертатися безпосередньо до бази даних, оминаючи важку логіку доменного рівня. Інтеграція MediatR та CQRS забезпечила створення «тонких контролерів» (Thin Controllers).

Вони позбавлені прямої бізнес-логіки і займаються лише прийомом HTTP-запитів, формуванням відповідних команд чи запитів, їх передачею в обробники та поверненням статус-кодів. Також у контролерах активно застосовується патерн Result (метод .Match()), який дозволяє ефективно опрацьовувати помилки та успішні результати без використання важких механізмів винятків (Exceptions), трансформуючи їх у стандартизовані API-відповіді.

Керування станом та доменні моделі Ядро системи Vector розроблене із суворим дотриманням концепції багатої доменної моделі (Rich Domain Model). На противагу анемічним моделям даних, тут усі параметри сутностей мають приватні модифікатори доступу для зміни (private set). Ініціалізація нових об'єктів виконується винятково через фабричний метод New, що забезпечує валідацію початкового стану. Модифікація стану об'єкта здійснюється лише через спеціальні поведінкові функції, такі як UpdateStatus(), Start(), Deactivate(). Це надійно захищає об'єкт від неконсистентних змін ззовні та зосереджує логіку його життєвого циклу всередині самої доменної моделі. Централізована перевірка всіх вхідних API-запитів виконується за допомогою FluentValidation, що гарантує цілісність даних до їхньої передачі в бізнес-середовище.

Шар доступу до інформації (Data Access Layer) Комунікація із СУБД PostgreSQL реалізована за допомогою патерну Repository та інструментарію Entity Framework Core. Класи-репозиторії виконують подвійну роль і імплементують два окремі інтерфейси: для операцій зчитування та для модифікації чи запису. З метою оптимізації швидкодії усі запити на отримання даних (Queries) застосовують метод .AsNoTracking(), що вимикає функцію відстеження змін у пам'яті сервера (Change Tracking), суттєво економлячи оперативну пам'ять під час обробки великих масивів інформації. Ще одним важливим архітектурним кроком є використання функціонального патерну Option<T> за допомогою бібліотеки Optional. Замість повернення значення null, яке часто спричиняє NullReferenceException, система повертає Option.Some() при успіху або Option.None, якщо запису немає. Це змушує розробників обробляти обидва варіанти на рівні компіляції, що підвищує загальну відмовостійкість (Fault Tolerance) платформи.

Ключові програмні підсистеми Функціонал бекенду поділено на низку ізольованих, але тісно взаємопов'язаних модулів. Блок автентифікації та профілювання реалізовано через відповідні контролери доступу. Він гарантує найвищий рівень захисту через використання протоколу OAuth 2.0 (Google OAuth). Після успішної валідації зовнішнього токена генерується внутрішній JWT, який захищає подальші звернення до ендпоінтів. У профілі клієнта (UserProfile) зберігаються критичні фізіологічні показники: вік, стать, зріст, вага, рівень підготовки, обраний спосіб життя, алергії та дієтичні обмеження. Центральним компонентом є модуль управління тренуваннями. Юзер відправляє запит із зазначенням мети, мови, графіка, наявного обладнання та виключених м'язів. Згенерований контент має ієрархічний вигляд: загальний макроцикл розбивається на дні (PlanDay), які містять списки вправ (PlanExercise) з бази довідників. Модуль нутриціології формує добові харчові цілі (DailyNutritionGoal), які синхронізуються зі спортивними днями. Кожен прийом їжі логується для порівняння із цільовими показниками та подальшої аналітики. Підсистема управління контентом забезпечує гнучке адміністрування. Зокрема, завантаження відеоінструкцій до вправ відбувається через потік даних (OpenReadStream()), що дозволяє безпечно передавати великі файли у хмарне сховище без перевантаження оперативної пам'яті сервера.

Асинхронні процеси та фонові завдання (Background Workers) З огляду на те, що генерація персоналізованого плану штучним інтелектом вимагає значних ресурсів і часу (від кількох секунд до хвилини), блокувати основний потік HTTP-запитів не можна. Тому в системі впроваджено механізм фонового опрацювання (Background Processing). Під час запиту ініціюється створення тимчасової сутності плану зі статусом «Creating». Завдання WorkoutPlanGenerationTask направляється у внутрішню чергу IWorkoutPlanGenerationQueue, а клієнт отримує статус HTTP 202 Accepted. Для відстеження готовності фронтенд здійснює поллінг (polling) спеціального ендпоінту. Тим часом фоновий сервіс WorkoutPlanGeneratorWorker, успадкований від BackgroundService, безперервно прослуховує чергу. Отримавши нове завдання, він

створює ізольовану область видимості, збирає потрібні дані з репозиторіїв та викликає AI-сервіс. При успішному завершенні план зберігається у базі, а статус змінюється на «Available». Якщо ж виникає виняткова ситуація, вмикається обробка помилок і статус переходить у «Failed», дозволяючи коректно повідомити користувача про збій.

Інтеграція з технологіями ШІ та Prompt Engineering Найбільш інноваційною частиною бекенду Vector є глибока взаємодія з генеративними нейромережами. Сервіс GeminiGenerativeService напряму комунікує з REST API Google Generative Language. Він підтримує як виключно текстові промпти, так і мультимодальні запити із зображеннями у форматі Base64, що критично важливо для візуального аналізу їжі. Особливу наукову цінність має реалізація алгоритмів промпт-інжинірингу. Архітектура запитів будується на основі системних інструкцій та динамічного контексту (Context Injection). Алгоритм формує детальний текст, передаючи нейромережі антропометрію, наявний інвентар та жорсткі технічні вимоги (зокрема повернення відповіді виключно у форматі JSON з використанням існуючих ідентифікаторів вправ, аби повністю уникнути галюцинацій моделі). Крім генерації планів, штучний інтелект виконує ще дві ключові функції:

1. Аналіз фотографій їжі (Meal Photo Analysis): користувач завантажує фотографію, а модель комп'ютерного зору (Vision AI) розпізнає страву, вираховує її калорійність і макроси, автоматично зберігаючи результат у щоденник.
2. Синтезатор страв (Meal Synthesizer): якщо у юзера залишилися невикористані калорії на кінець дня, алгоритм формує запит до ШІ, який генерує персональний рецепт, що ідеально закриває добову норму без перевищення лімітів. Усі процеси комунікації зі штучним інтелектом відбуваються виключно асинхронно на серверному боці, гарантуючи високу швидкість обробки та абсолютну безпеку API-ключів.

3.4. Розробка клієнтської частини (Frontend)

Розробка користувацького інтерфейсу (Frontend) є одним із найважливіших етапів створення системи Vector, оскільки саме клієнтська частина безпосередньо взаємодіє з кінцевим користувачем, формуючи його враження від продукту та забезпечуючи зручний доступ до складного інтелектуального функціоналу. З огляду на специфіку проекту — необхідність обробки великих масивів даних (тренувальні плани, списки вправ, щоденники харчування) та динамічної взаємодії з AI-модулями — архітектуру фронтенду було побудовано за принципом односторінкового застосунку (Single Page Application, SPA).

В якості базової технології було обрано бібліотеку React у поєднанні зі строго типізованою мовою TypeScript. Використання TypeScript на клієнтській стороні є критично важливим рішенням: воно забезпечує повну відповідність доменним моделям бекенду (наприклад, складним ієрархіям `WorkoutPlan` або `SynthesizedMeal`), унеможливлює виникнення помилок, пов'язаних з невідповідністю типів даних під час виконання програми, та значно спрощує рефакторинг коду. Для забезпечення високої швидкості локальної розробки та оптимізованої збірки проекту для робочого середовища (production) використовується сучасний інструмент Vite.

Керування глобальним станом застосунку (State Management) реалізовано на базі архітектури Redux з використанням сучасного набору інструментів Redux Toolkit (RTK). Це дозволяє централізовано зберігати сесійні дані користувача, токени авторизації, поточні статуси генерації планів та результати аналізу фотографій їжі. Завдяки RTK мінімізовано кількість шаблонного коду (boilerplate), а логіка взаємодії з серверним API ізольована в окремі асинхронні екшени (Thunks).

Особлива увага під час розробки приділялася UX/UI рішенням (User Experience / User Interface). Візуальний стиль платформи побудовано на основі сучасної, чистої та мінімалістичної темної теми (Dark Theme). Темний інтерфейс не лише відповідає сучасним трендам у проектуванні застосунків для фітнесу, але й знижує навантаження на зір користувача під час щоденного використання. Для стилізації компонентів імплементовано utility-first фреймворк Tailwind CSS. Такий

підхід дозволив створити гнучку, повністю адаптивну (Responsive) дизайн-систему, яка коректно відображається на екранах будь-якого розміру — від широкоформатних моніторів до мобільних пристроїв, без необхідності написання сотень рядків кастомного CSS-коду. Інтерфейс підтримує багатомовність (реалізовано через i18next), що робить платформу доступною для користувачів з України, Польщі та англомовних країн.

Адаптивний дизайн та кросплатформність

З огляду на специфіку предметної області (фітнес та нутриціологія), передбачається, що значна частина користувачів взаємодіятиме з платформою Vector через мобільні пристрої — безпосередньо під час виконання вправ у тренажерному залі або для оперативного фотографування страв. Саме тому архітектуру користувацького інтерфейсу було спроектовано з дотриманням парадигми Mobile-First (мобільні пристрої перш за все). Цей підхід передбачає, що розробка та оптимізація інтерфейсу першочергово орієнтувалася на екрани смартфонів, з подальшим поступовим розширенням та ускладненням макету (Progressive Enhancement) для планшетів і широкоформатних десктопних моніторів.

Технічна реалізація адаптивності (Responsive Design) повністю покладена на гнучку систему медіа-запитів (media queries), вбудовану у фреймворк Tailwind CSS. Використання стандартизованих точок зупину (breakpoints) дозволило створити динамічну «гумову» сітку макета на базі технологій Flexbox та CSS Grid. Завдяки цьому складні багатоколонкові структури, які відображаються на екранах комп'ютерів, автоматично трансформуються в ергономічну одноколонкову структуру на мобільних пристроях. Крім того, адаптація охоплює зміну розмірів типографіки, оптимізацію відступів (padding/margin) для зручного керування сенсорним екраном (Touch Targets) та трансформацію навігаційних панелей у компактні приховані меню. Такий підхід гарантує цілісний, безшовний користувацький досвід (UX) та збереження 100% функціоналу системи незалежно від діагоналі екрана чи операційної системи пристрою.

Головна сторінка (Home Page)

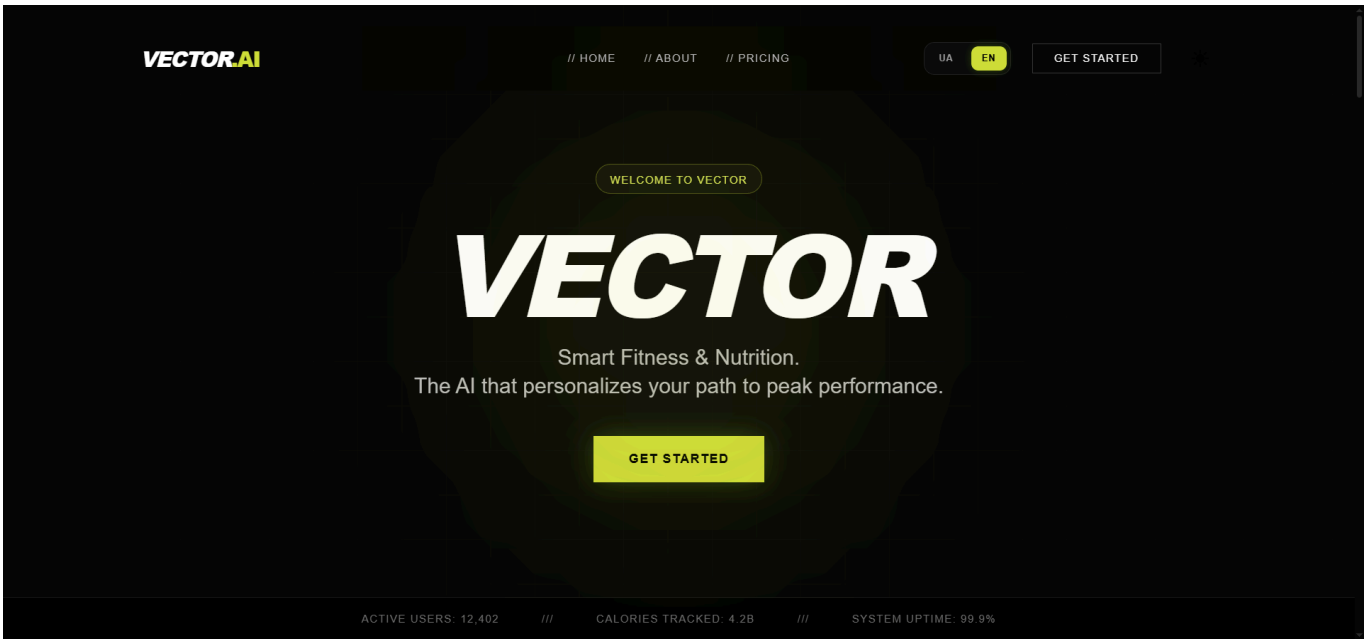

Рис. 3.1. Скріншот першого екрана (Hero Section)

Джерело: розроблено автором

Головна сторінка (Home Page) слугує ключовою точкою входу до системи та виконує роль презентаційного дашборду, що розкриває основну цінність продукту. Головне завдання цієї сторінки з точки зору UX/UI — миттєво донести до користувача концепцію Vector як персонального AI-тренера та дієтолога, уникнувши при цьому перевантаження інтерфейсу зайвою інформацією.

Візуальна ієрархія першого екрана (Hero Section) побудована навколо центрального повідомлення, яке чітко артикулює головну перевагу платформи. Дизайн цього блоку виконано у стриманій кольоровій гамі глибокої темної теми (колір фону #050505) з використанням неоново-лаймових акцентів (Tailwind-клас lime-500). Контрастна кнопка цільової дії (Call to Action — CTA) інтуїтивно спонукає нового користувача розпочати процес генерації власного плану. Верхня навігаційна панель містить зручний перемикач локалізації (UA/EN), реалізований за допомогою бібліотеки react-i18next, що забезпечує миттєву зміну мови без перезавантаження сторінки. У нижній частині екрана реалізовано інформаційну панель (Stats Bar), яка відображає поточну статистику системи (кількість активних користувачів, відстежені калорії), що підвищує рівень довіри до продукту.

З точки зору програмної реалізації, Головна сторінка побудована за принципом компонентної архітектури. Весь код декомпоновано на незалежні функціональні модулі (Functional Components), такі як SectionHeader, BentoItem, UserStatCard та PricingCard. Це не лише відповідає парадигмі DRY (Don't Repeat Yourself), але й значно спрощує підтримку та тестування інтерфейсу. Для створення динамічного користувацького досвіду інтегровано бібліотеку анімацій framer-motion. Завдяки хукам whileInView та viewport={{ once: true }}, елементи сторінки плавно з'являються під час прокручування (Scroll Animations) з налаштованими затримками (staggered delay), що додає інтерфейсу плавності та сучасності.

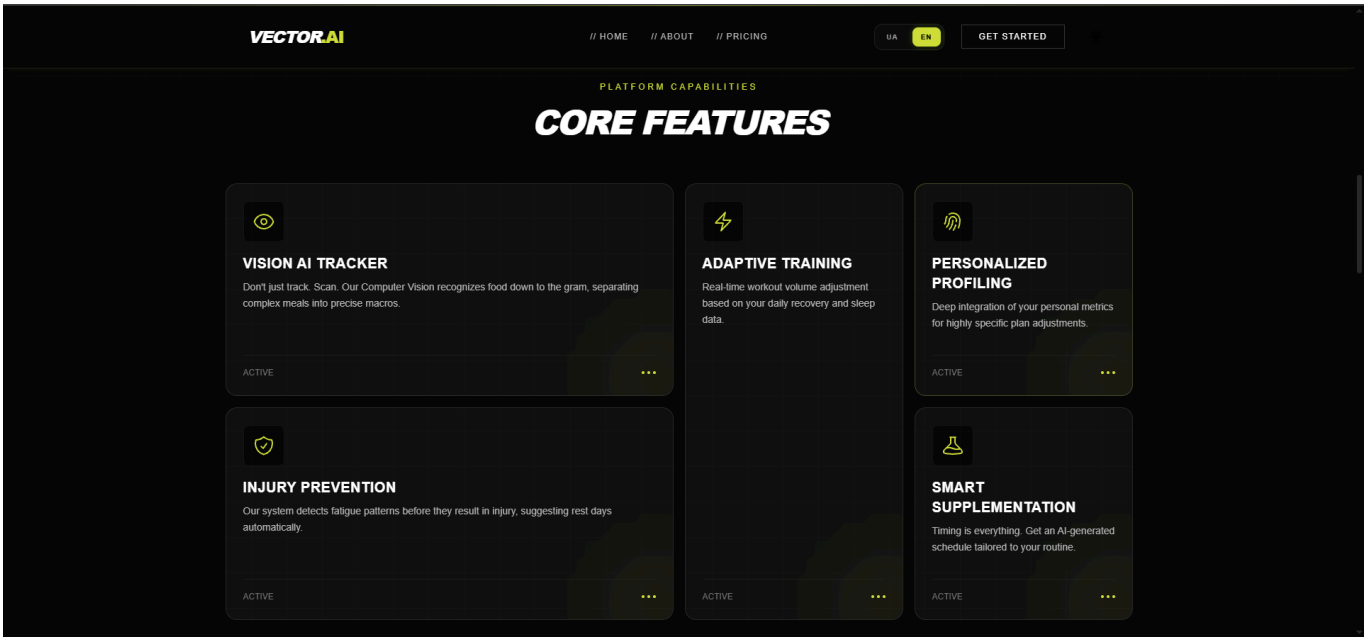

Рис. 3.2. Скріншот секції з Bento-картками

Джерело: розроблено автором

Окремої уваги заслуговує реалізація інформаційних блоків та їхня адаптивність до різних пристроїв. Секція архітектури платформи реалізована з використанням популярного в сучасному веб-дизайні патерну Bento Grid. Компонент BentoItem поєднує в собі іконки (з використанням @heroicons/react), текстову інформацію та анімовані індикатори статусу (пульсуючі точки), створюючи ефект "живої" системи.

У контексті мобільної адаптації, код Головної сторінки яскраво демонструє підхід Mobile-First. Використання сіткових класів Tailwind CSS, таких як grid-cols-1 md:grid-cols-3 lg:grid-cols-4, гарантує, що на мобільних телефонах усі складні багатоколонкові структури (кроки алгоритму, тарифні плани, відгуки користувачів) автоматично трансформуються у вертикальний список (одна колонка). Зі збільшенням ширини екрана (на планшетах та десктопах) інтерфейс плавно перебудовується, максимально ефективно використовуючи доступний вільний простір, зберігаючи при цьому задані пропорції та читабельність тексту незалежно від роздільної здатності пристрою користувача.

Сторінка «Про систему» (About Page)

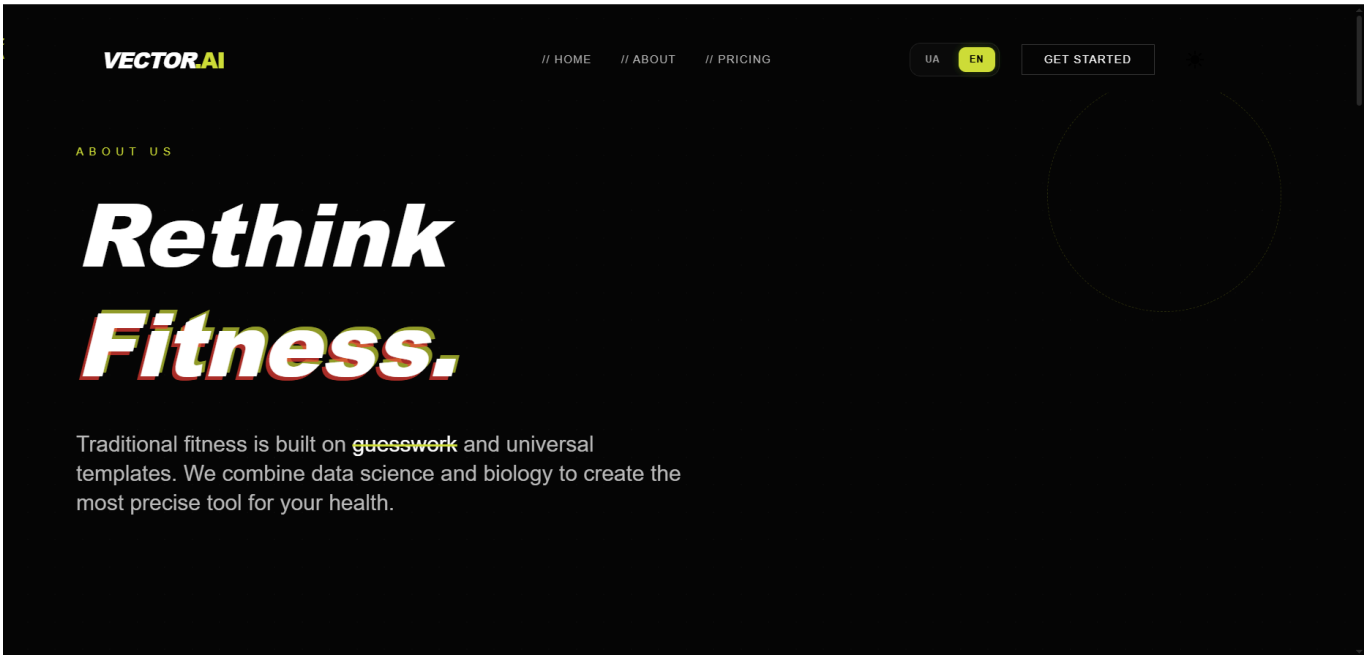

Рис. 3.3. Головний екран сторінки «About Us»

Джерело: розроблено автором

Сторінка «Про систему» (About Page) виконує важливу репутаційну та освітню функцію в архітектурі клієнтської частини застосунку Vector. Її головна мета — розкрити технологічну філософію продукту, етапи його створення та ключові цінності, формуючи у користувача довіру до штучного інтелекту як до надійного

фітнес-асистента. Візуальна концепція цієї сторінки продовжує загальний мінімалістичний стиль платформи, проте відрізняється підвищеним рівнем інтерактивності та застосуванням складних анімаційних патернів.

Однією з унікальних UI-знахідок є інтеграція компонента SystemHUD — фіксованої бічної панелі, що виконує роль візуального індикатора прогресу читання сторінки. З технічної точки зору, цей елемент безпосередньо зв’язаний із хуком useScroll бібліотеки framer-motion. Завдяки використанню фізичної моделі пружини (useSpring), заповнення індикатора відбувається максимально плавно, без різких стрибків під час швидкого скролінгу. Важливо відзначити застосування адаптивного підходу: цей елемент має CSS-класи hidden md:flex, що означає його повне приховування на екранах мобільних пристроїв. Це класичний приклад оптимізації UX для смартфонів, де кожен піксель горизонтального простору є критично важливим, і другорядні декоративні елементи не повинні заважати читанню основного контенту.

Заголовки сторінки реалізовані через кастомний компонент GlitchHeader, який використовує багатошаровість тексту та CSS-транзиції для створення візуального ефекту зсуву кольорових каналів (RGB-split) при наведенні курсору. Фонове оформлення секцій побудоване на основі радіальних градієнтів та патернів (наприклад, крапкова сітка radial-gradient), що створює ефект глибини простору без використання важких растрових зображень, позитивно впливаючи на швидкість завантаження (Performance) клієнтського застосунку.

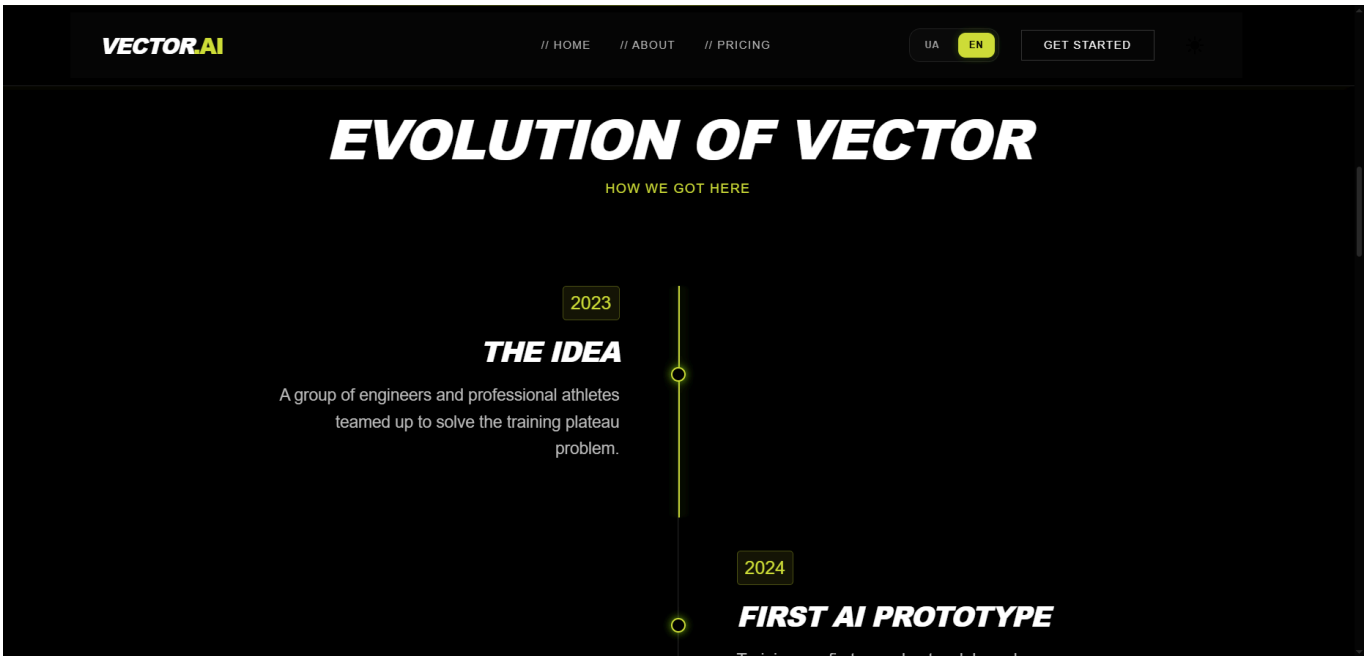

Рис. 3.4. Інтерактивна шкала етапів розвитку та роботи алгоритму (Vertical Timeline)

Джерело: розроблено автором

Для подання хронологічної інформації або послідовності дій алгоритму розроблено складний компонент VerticalTimeline. Цей блок використовує Intersection Observer API (через хук whileInView), щоб активувати анімацію появи кожного окремого кроку лише тоді, коли користувач докручує до нього сторінку (viewport-triggered animations). Центральна візуальна вісь таймлайну також динамічно заповнюється кольором (стиль scaleY: scrollYProgress), синхронізуючись із рухом користувача.

У цьому компоненті бездоганно реалізовано мобільну адаптацію (Responsive Design). На десктопних моніторах контент чергується у шаховому порядку зліва та справа від центральної лінії завдяки умовним класам (isEven ? 'md:flex-row-reverse' : ''). Проте на екранах мобільних телефонів така структура була б нечитабельною. Тому Tailwind-конфігурація автоматично перебудовує таймлайн: центральна вісь зміщується до лівого краю екрана (left-4 md:left-1/2), а весь текст вирівнюється по лівому боці з відповідними відступами (pl-12 md:pl-0). Це гарантує, що інформація залишається структурованою і легко сприймається на пристроях з будь-якою діагоналлю.

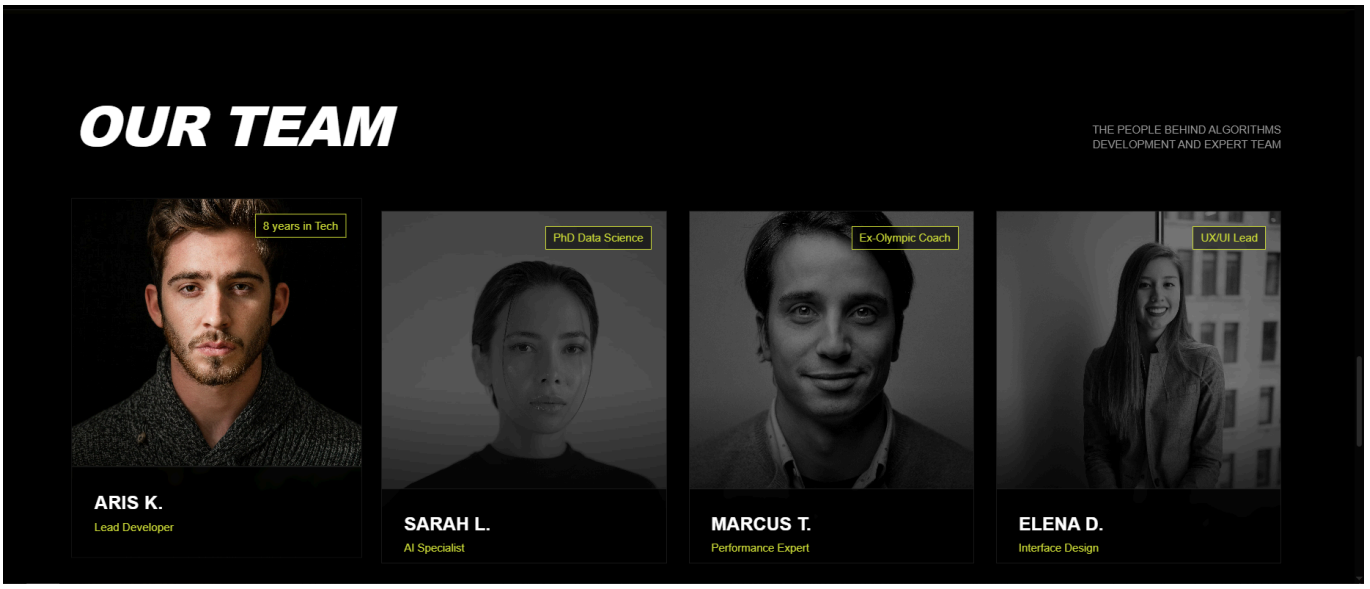

Рис. 3.5. Блок представлення команди

Джерело: розроблено автором

Інформація про команду розробників або експертів виведена у вигляді суворої сіткової структури, побудованої за допомогою CSS Grid (grid-cols-1 md:grid-cols-2 lg:grid-cols-4). Картки співробітників (DataCard) інкапсулюють у собі складну логіку роботи із зображеннями. За замовчуванням фотографії мають знижену прозорість та накладений фільтр знебарвлення (grayscale), що дозволяє їм не вибиватися із загальної темної колірної схеми сайту. При взаємодії з користувачем (Hover) зображення плавно набувають кольору, а блок з інформацією піднімається знизу (translate-y-4 group-hover:translate-y-0).

Уся текстова інформація на сторінці, включаючи описи цінностей, імена, посади та пункти таймлайну, генерується динамічно через словники інтернаціоналізації (бібліотека react-i18next та хук useTranslation). Замість жорстко закодованого тексту (hardcoded text) використовуються ключі локалізації (наприклад, t('about.team.member1.name')), що робить компонент повністю незалежним від обраної мови та готовим до подальшого масштабування платформи на нові регіональні ринки. Завершується сторінка контрастним світлим блоком (Footer CTA), який психологічно перемикає увагу користувача та пропонує дві цільові дії: ініціалізацію платформи або перехід до документації.

Сторінка обробки помилки навігації (404 Not Found)

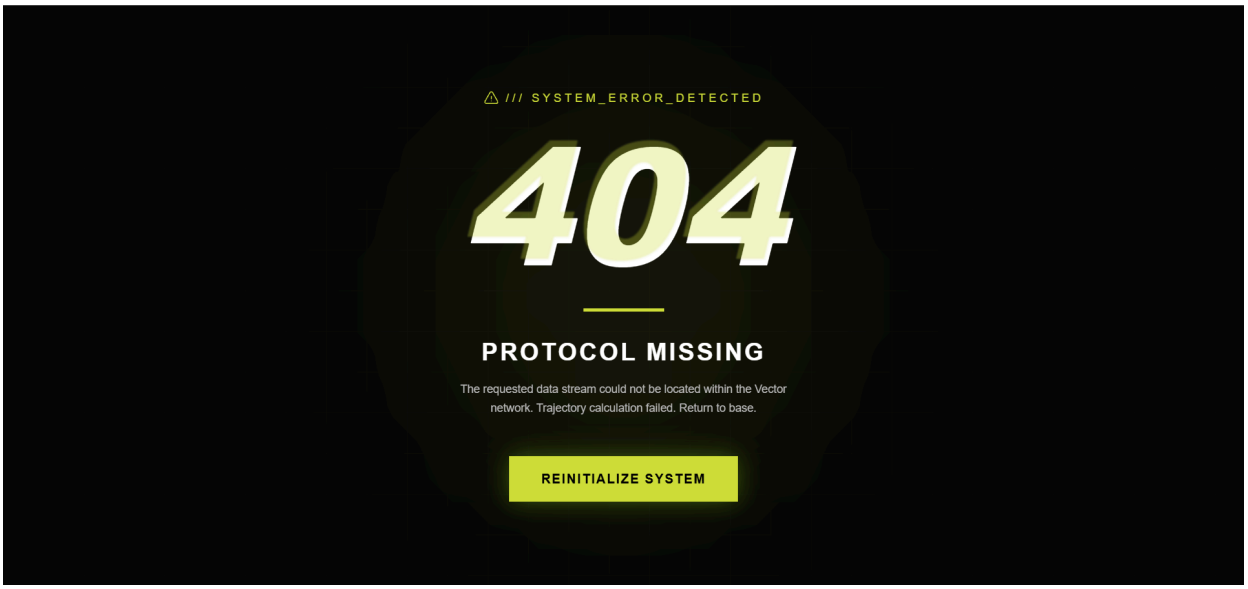

Рис. 3.6. Кастомна сторінка помилки 404

Джерело: розроблено автором

Невід'ємною складовою будь-якого сучасного вебзастосунку, побудованого за архітектурою SPA (Single Page Application), є надійна та коректна обробка неіснуючих маршрутів. Оскільки у React-застосунках маршрутизація відбувається на стороні клієнта (Client-Side Routing) за допомогою бібліотеки React Router, спроба користувача перейти за неправильним або застарілим посиланням не повинна призводити до стандартної «білої сторінки» браузера. Для вирішення цього завдання у системі Vector розроблено кастомну сторінку помилки 404 (Not Found Page), яка перехоплює всі невідомі URL-адреси (через маршрут типу path="*") та елегантно повертає користувача в робоче середовище програми.

З точки зору UX/UI-дизайну, сторінка 404 у системі Vector є чудовим прикладом збереження цілісності бренду (Brand Consistency) та використання мікрокопірайтингу для покращення користувацького досвіду. Замість сухого технічного повідомлення про помилку, розробники застосували елементи гейміфікації, інтегрувавши стилістику наукової фантастики та кібернетичних систем, що ідеально резонує з позиціонуванням продукту як штучного інтелекту. Це простежується у використанні моноширинних шрифтів та специфічної термінології: попереджувальний маркер /// SYSTEM_ERROR_DETECTED, заголовок PROTOCOL

MISSING та повідомлення про неможливість розрахунку траєкторії («Trajectory calculation failed. Return to base»).

Візуальна складова побудована на глибокому чорному фоні екрана з яскравими лаймовими (neon-lime) акцентами. Головний елемент — цифри «404» — реалізовано з використанням багаторівневих текстових тіней (Drop Shadow) та зміщення, що створює об'ємний ефект світіння або так званий ефект контрольованого глітчу (glitch). Нижче розташована масивна кнопка цільової дії (Call to Action) з написом REINITIALIZE SYSTEM, яка супроводжується ефектом зовнішнього світіння (box-shadow). Програмно ця кнопка прив'язана до хука навігації (useNavigate() з React Router), що дозволяє миттєво перенаправити користувача на головну сторінку без перезавантаження браузера, зберігаючи поточний стан застосунку (Redux State).

Важливим аспектом реалізації цієї сторінки є її бездоганна мобільна адаптація. Завдяки використанню технології Flexbox (flex flex-col items-center justify-center), увесь контент завжди ідеально відцентровано як по вертикалі, так і по горизонталі (Absolute Center), незалежно від висоти екрана пристрою. За допомогою адаптивних класів Tailwind CSS налаштовано динамічне масштабування типографіки: масивний шрифт цифр «404» та заголовка плавно зменшується на екранах смартфонів, щоб уникнути появи горизонтального прокручування. Крім того, на мобільних пристроях кнопка повернення оптимізує свою ширину для зручного натискання пальцем (Touch Target Optimization), гарантуючи, що навіть потрапивши на неіснуючу сторінку зі смартфона, користувач зможе інтуїтивно та швидко повернутися до використання платформи.

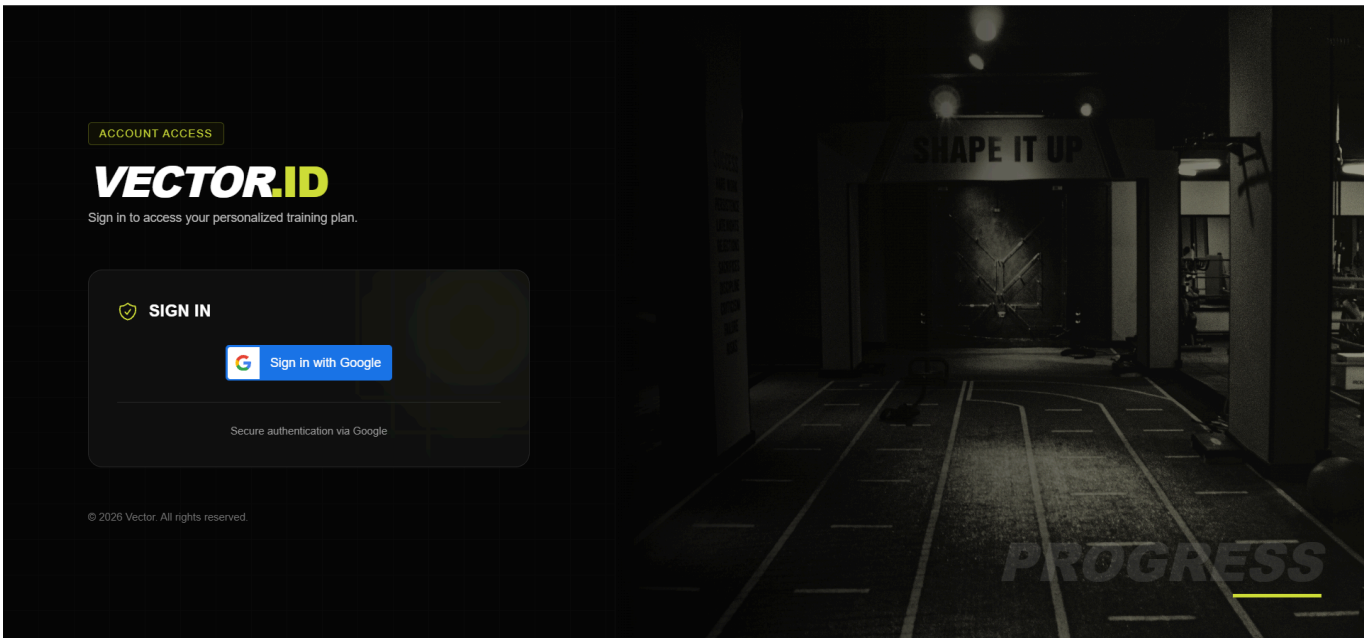

Рис. 3.7. Сторінка авторизації

Джерело: розроблено автором

Сторінка авторизації призначена для безпечного доступу користувачів до персонального середовища платформи. Для спрощення процесу входу та підвищення рівня безпеки замість традиційної реєстрації з паролем тут реалізовано сучасний механізм авторизації виключно через обліковий запис Google.

Візуально інтерфейс сторінки розділений на дві основні частини, що є класичним патерном для сучасних вебзастосунків. Ліва частина містить безпосередньо форму входу: привітальний заголовок, короткий опис системи та інтерактивну кнопку для виклику вікна Google авторизації. Права частина, яка відображається лише на достатньо широких екранах, виконує декоративну функцію, демонструючи стилізоване фонове зображення з ефектом накладання фірмових кольорів та анімованим водяним знаком системи.

Програмна логіка цієї сторінки відповідає не лише за виклик вікна авторизації, але й за обробку отриманого від сервера результату. Після успішного входу система отримує зашифрований токен доступу, локально розкодовує його для зчитування інформації про роль користувача і виконує автоматичне перенаправлення. Якщо система виявляє права адміністратора, відбувається перехід до панелі управління системою, в іншому випадку звичайний користувач потрапляє на свій персональний

дашборд. Усі елементи інтерфейсу з'являються плавно завдяки попередньо налаштованим анімаціям, що зберігає загальну динаміку роботи платформи.

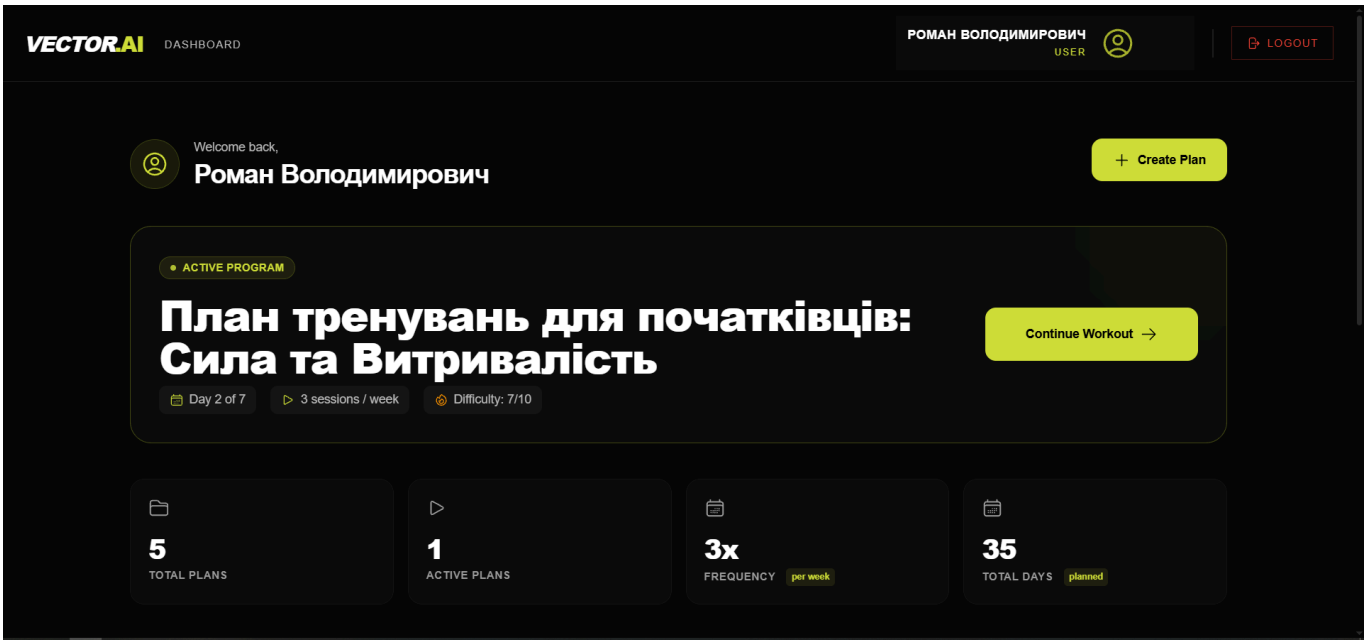

Рис. 3.8. Головна інформаційна панель

Джерело: розроблено автором

Дашборд (головна інформаційна панель) є центральним вузлом клієнтської частини системи Vector, куди користувач потрапляє одразу після успішної авторизації. Головне завдання цієї сторінки полягає в агрегації ключових даних про поточний тренувальний процес, фізичні показники користувача та наданні швидкого доступу до основного функціоналу платформи. Архітектура сторінки побудована за принципом адаптивної модульної сітки, що дозволяє ефективно розмістити великий обсяг різнотипної інформації без візуального перевантаження інтерфейсу.

Верхня частина екрана містить персоналізоване привітання та акцентну кнопку для ініціалізації процесу створення нового персонального плану. Якщо у користувача вже є активна тренувальна програма, система автоматично виводить її у вигляді великої інформаційної картки. На цій картці відображається назва макроциклу, розрахований поточний день тренувань, частота занять на тиждень та загальний рівень складності. Це дозволяє користувачу миттєво продовжити своє тренування

натисканням однієї кнопки, без необхідності шукати потрібний план у глибоких рівнях навігації.

Під блоком активної програми розташована панель динамічної статистики, яка складається з чотирьох карток. Для оптимізації продуктивності клієнтського застосунку розрахунок цих показників (загальна кількість згенерованих планів, кількість активних програм, середня частота тренувань та сумарна тривалість у днях) виконується безпосередньо на стороні клієнта за допомогою хука мемоізації. Це гарантує, що складні обчислення масиву даних відбуваються лише при фактичній зміні списку програм, що ефективно запобігає зайвим перемальовуванням інтерфейсу (ререндерам) та економить ресурси пристрою.

Важливою архітектурною та технічною особливістю дашборду є реалізація механізму асинхронного опитування сервера (long polling). Оскільки генерація персонального плану штучним інтелектом є ресурсоємним процесом і займає певний час, на сторінці імплементовано фоновий процес, який із заданим інтервалом перевіряє статус генерації. Якщо бекенд повідомляє про успішне завершення формування плану, система автоматично зупиняє цикл опитування та оновлює сторінку для відображення нових даних. Якщо ж AI-сервіс повертає помилку, інтерфейс перехоплює цей стан і виводить відповідне сповіщення. Це забезпечує безперебійний користувацький досвід, дозволяючи системі працювати у фоновому режимі без блокування взаємодії з інтерфейсом.

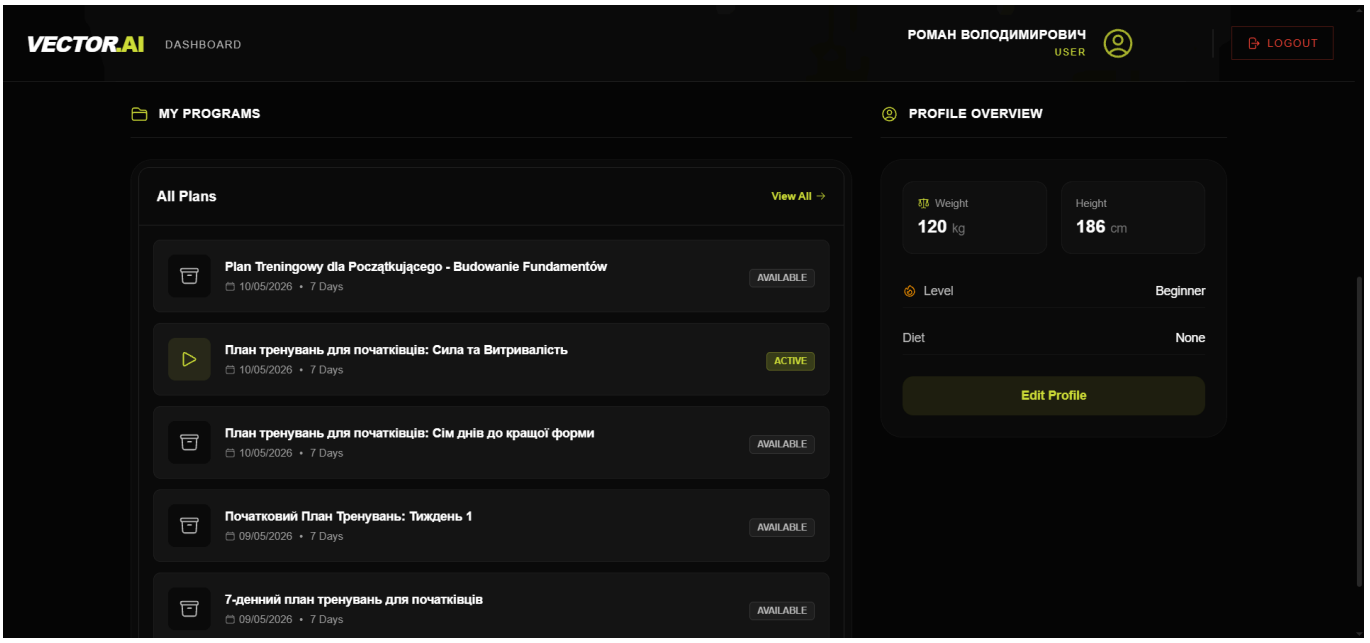

Рис. 3.9. Розділення інтерфейсу дашборду на архів програм та панель огляду параметрів користувача

Джерело: розроблено автором

Нижня частина дашборду структурно розділена на дві логічні колонки з використанням системи CSS Grid. Ліва, ширша колонка, містить інтерактивний список усіх створених програм користувача з відображенням їхніх статусів. Права колонка відведена під панель швидкого огляду профілю. Тут у компактному вигляді відображаються актуальні антропометричні дані, зокрема поточна вага та зріст, а також обраний рівень підготовки і тип дієти.

Логіка компонента передбачає перевірку повноти введених даних: якщо профіль користувача не заповнений до кінця, система замість параметрів виводить попереджувальний блок з іконкою та пропонує швидкий перехід до сторінки налаштувань. Завдяки використанню глобального стану інструменту управління Redux та ізольованих сервісних функцій для запитів до API, сторінка миттєво та коректно реагує на зміни даних, забезпечуючи високу швидкість роботи платформи та плавність появи елементів інтерфейсу.

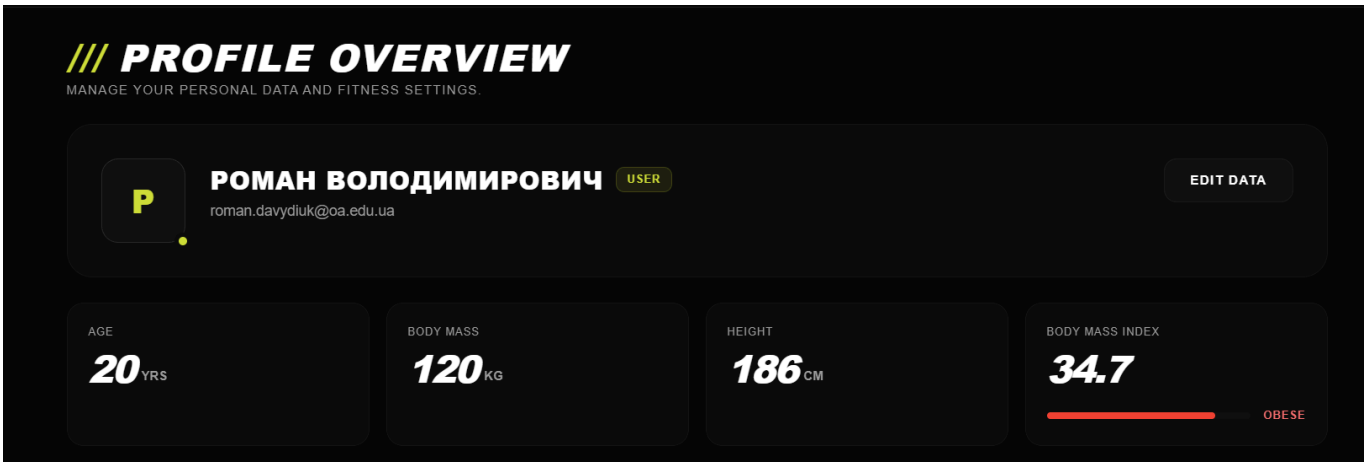

Рис. 3.10. Панель управління біометричними даними користувача

Джерело: розроблено автором

Сторінка профілю користувача є критично важливим модулем системи Vector, оскільки саме тут збираються та зберігаються антропометричні дані, необхідні для коректної роботи алгоритмів штучного інтелекту при генерації персоналізованих планів тренувань і харчування. Архітектурно сторінка побудована на основі локального стану керування відображенням, який перемикає інтерфейс між трьома режимами: основним інформаційним дашбордом, майстром первинного налаштування (Wizard) та формою редагування існуючих даних. У випадку, якщо новий користувач ще не ввів свої біометричні параметри, система автоматично приховує порожні графіки і виводить акцентний блок з вимогою заповнити анкету, гарантуючи таким чином наявність базового контексту для подальшої коректної роботи AI-сервісів.

Особливістю інформаційного дашборду профілю є автоматична обробка та візуалізація даних безпосередньо на стороні клієнта. Замість зберігання статичного віку чи індексу маси тіла (ІМТ) у базі даних, клієнтський застосунок динамічно вираховує поточний вік на основі дати народження та обчислює ІМТ за класичною формулою співвідношення ваги до квадрата зросту. Результат розрахунку ІМТ проходить через вбудований алгоритм категоризації, який не лише повертає медичне трактування показника (наприклад, недостатня вага, нормальна або надлишкова), але й автоматично генерує відповідний колірний індикатор (від синього до червоного). Цей індикатор інтегровано у горизонтальну шкалу прогресу, що дозволяє

користувачу інтуїтивно та швидко оцінити свій поточний фізичний стан без необхідності заглиблюватися в цифри.

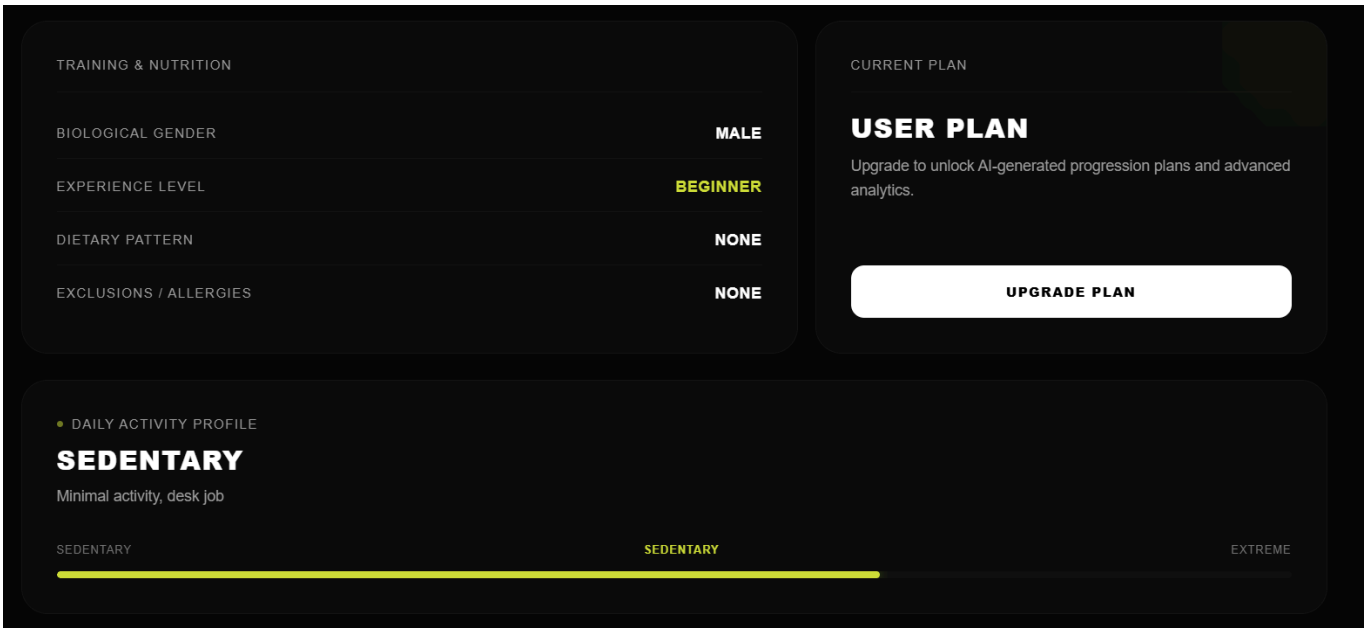

Рис. 3.11. Деталізація тренувальних налаштувань

Джерело: розроблено автором

Нижня частина сторінки профілю деталізує специфічні налаштування користувача, розподіляючи їх на чіткі логічні блоки. Секція тренувань та харчування у вигляді мінімалістичного списку виводить інформацію про біологічну стать, рівень тренувального досвіду, обраний тип дієти та наявні харчові алергії чи виключення з раціону. Поруч розташований блок поточного тарифного плану з інформацією про підписку та інтерфейсом для її оновлення.

Окрему увагу приділено секції щоденної активності (Daily Activity Profile), яка візуалізує обраний користувачем тип способу життя. Замість звичайного тексту тут імплементовано кастомний графічний індикатор — шкалу інтенсивності від малорухливого (Sedentary) до екстремального (Extreme) рівня. Відсоток заповнення цієї шкали обчислюється математично на основі індексу поточного способу життя відносно загальної кількості доступних опцій у базі даних системи. Уся сторінка та її окремі компоненти плавно з'являються на екрані з використанням каскадних затримок анімації, що створює футуристичний користувацький досвід, а переходи

між режимами перегляду та редагування відбуваються без перезавантаження сторінки завдяки алгоритмам збереження візуального контексту.

Інтерфейс конструювання персонального плану (Workout Wizard)

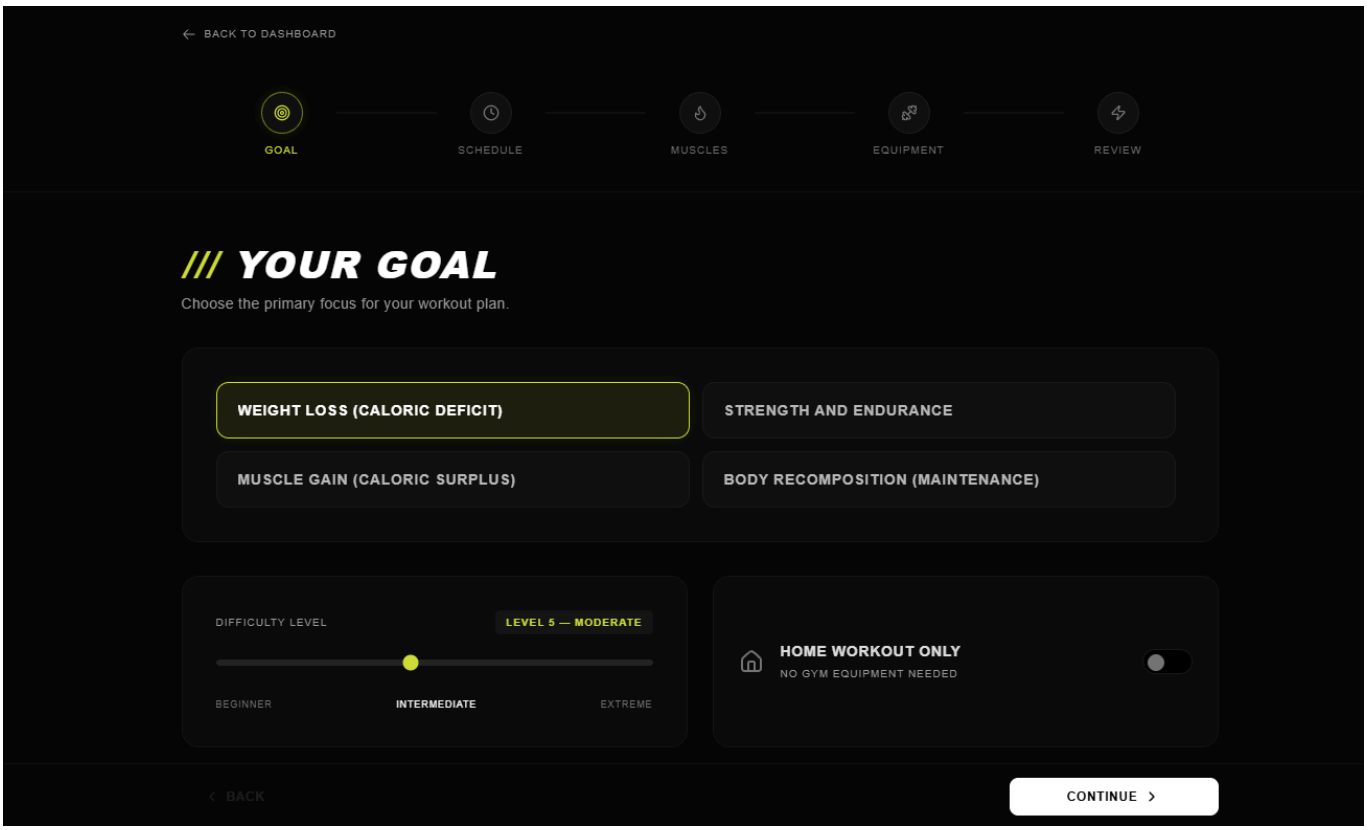

Рис. 3.12. Перший крок інтерактивного майстра створення плану

Джерело: розроблено автором

Процес створення нового тренувального плану є ключовим етапом взаємодії користувача з експертною системою Vector. Оскільки для генерації якісного результату штучному інтелекту потрібен максимально точний та об'ємний контекст (мета, графік, цільові м'язи, наявне обладнання), виведення всіх цих налаштувань на одну сторінку призвело б до когнітивного перевантаження користувача (Cognitive Overload). Для вирішення цієї проблеми інтерфейс конструювання плану реалізовано у вигляді багатокрокового інтерактивного майстра (Wizard Pattern). Такий підхід розбиває складний процес на п'ять логічних, послідовних кроків, кожен з яких супроводжується візуальним індикатором прогресу (StepIndicator) у верхній частині екрана.

На першому кроці (зображеному на рисунку) користувач визначає фундаментальні параметри майбутнього макроциклу. Інтерфейс містить список цілей (Objectives), який завдяки інтегрованій функції пошуку легко фільтрується в режимі реального часу. Для налаштування інтенсивності реалізовано кастомний повзунок (Range Slider) зі шкалою від 1 до 10. Особливістю цього елемента є динамічна зміна кольорового кодування та текстового опису залежно від обраного значення (від нейтрального сірого для рівня «Beginner» до агресивного червоного для рівня «Extreme»). Також на цьому етапі розміщено перемикач формату тренувань «Home Workout Only».

Архітектура компонента відзначається високою гнучкістю та адаптивною логікою маршрутизації (Dynamic Routing). Наприклад, якщо користувач активує режим домашніх тренувань, система розуміє, що доступ до тренажерного залу відсутній, і програмно виключає крок вибору обладнання із загального ланцюжка візарда (змінюючи масив activeSteps). На кроках вибору цільових м'язів та інвентарю розроблено унікальний компонент селектора (TagSelector), який підтримує три стани вибору: фокус (обов'язкове включення в план), виключення (сувора заборона на використання, що критично важливо при травмах) та нейтральний стан (система сама вирішує доцільність використання).

Навігація між кроками супроводжується плавними просторовими транзиціями, реалізованими за допомогою бібліотеки framer-motion. Використання компонента AnimatePresence у поєднанні з відстеженням напрямку руху (змінна direction) дозволяє сторінкам природно «виїжджати» зліва чи справа, імітуючи поведінку нативних мобільних застосунків.

Останній крок візарда (Review) виконує функцію фінальної верифікації. Система збирає всі введені дані (State) і формує з них єдине структуроване досьє (Payload), яке зручно згруповане за категоріями. Лише після підтвердження на цьому етапі клієнтська частина відправляє запит на бекенд і через Redux-екшен startGeneratingPlan ініціює асинхронний процес генерації плану штучним інтелектом.

Адаптивність (Mobile-First) цього складного інтерфейсу забезпечується використанням фіксованої нижньої панелі навігації (fixed bottom-0). Незалежно від того, наскільки довгим є список м'язів чи інвентарю на конкретному кроці, кнопки «Назад» та «Продовжити» завжди залишаються в зоні доступу великого пальця користувача на смартфоні. Додатково застосовано оптимізацію скролбарів (кастомні стилі ::-webkit-scrollbar) для блоків із пошуком, що робить інтерфейс охайним і зручним на пристроях з будь-якою роздільною здатністю екрана.

Сторінка детального перегляду та виконання тренувального плану (Workout Details)

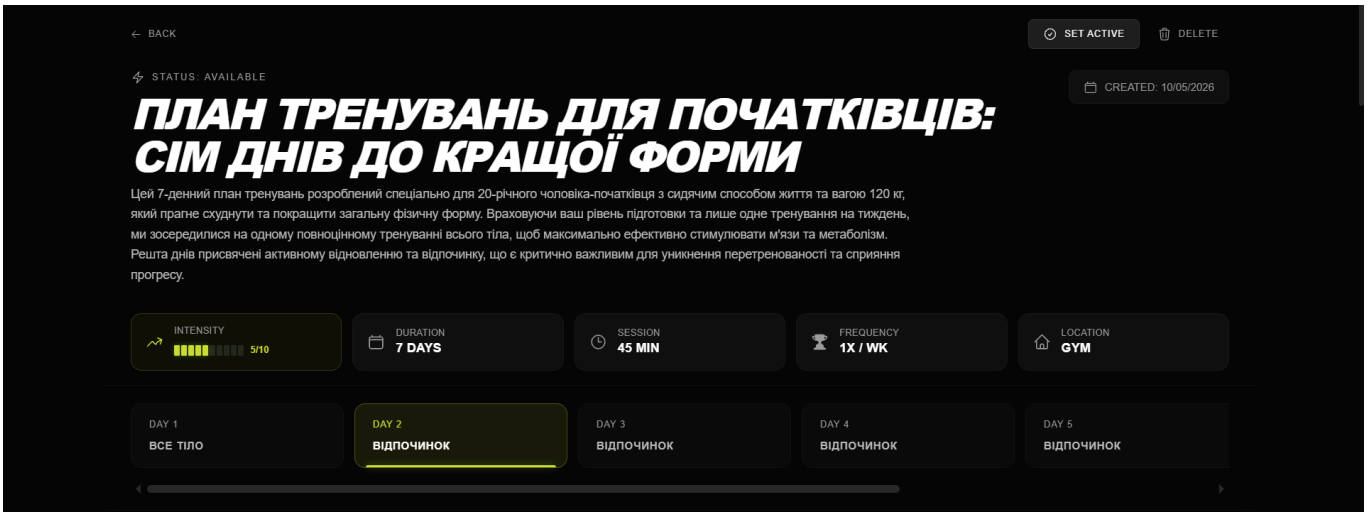

Рис. 3.13. Інформаційна панель поточного макроциклу

Джерело: розроблено автором

Сторінка детального перегляду тренувального плану є ключовим операційним середовищем платформи Vector, де користувач безпосередньо взаємодіє зі згенерованим контентом. Архітектурною особливістю цієї сторінки є об'єднання двох невід'ємних складових фітнес-прогресу — фізичних навантажень та нутриціології — в єдиний, безшовний інтерфейс. Це дозволяє користувачу не перемикатися між різними розділами застосунку під час щоденної рутини, маючи перед очима повну картину свого дня.

Верхня частина екрана (Header) виконує функцію загального інформаційного зведення. Тут розміщено назву програми, її поточний статус (наприклад, «Active» або «Available») та панель керування життєвим циклом плану (кнопки активації або перманентного видалення з підтвердженням через модальне вікно). Нижче розташована сітка аналітичних карток (Stat Cards), які візуалізують базові параметри макроциклу: розраховану алгоритмом інтенсивність (у вигляді кастомного десятирівневого індикатора), загальну тривалість, тривалість однієї сесії, частоту тренувань та цільову локацію. Для зручної навігації по днях розроблено горизонтальну стрічку з фіксацією (sticky-позиціонування) та підтримкою сенсорного прокручування. Алгоритм системи автоматично вираховує поточну дату відносно дня старту програми та динамічно маркує дні як «минулі», «сьогоднішні» або «майбутні», фокусуючи увагу користувача на актуальному завданні.

Нижня частина інтерфейсу розділена на дві взаємодіючі зони. Ліва колонка (яка на мобільних пристроях перетворюється на першу вкладку) присвячена безпосередньо вправам обраного дня. Кожна вправа представлена у вигляді мінімалістичної картки, що містить назву, згенеровані штучним інтелектом контекстні поради щодо виконання (AI Notes), а також чіткі параметри підходів, повторень та часу на відпочинок. При взаємодії з карткою система викликає висувну бічну панель (Exercise Drawer) з детальною технікою виконання. У випадку, якщо алгоритм призначив день відновлення, інтерфейс адаптується і виводить спеціальний блок «Rest Day» з рекомендаціями щодо відпочинку та гідратації.

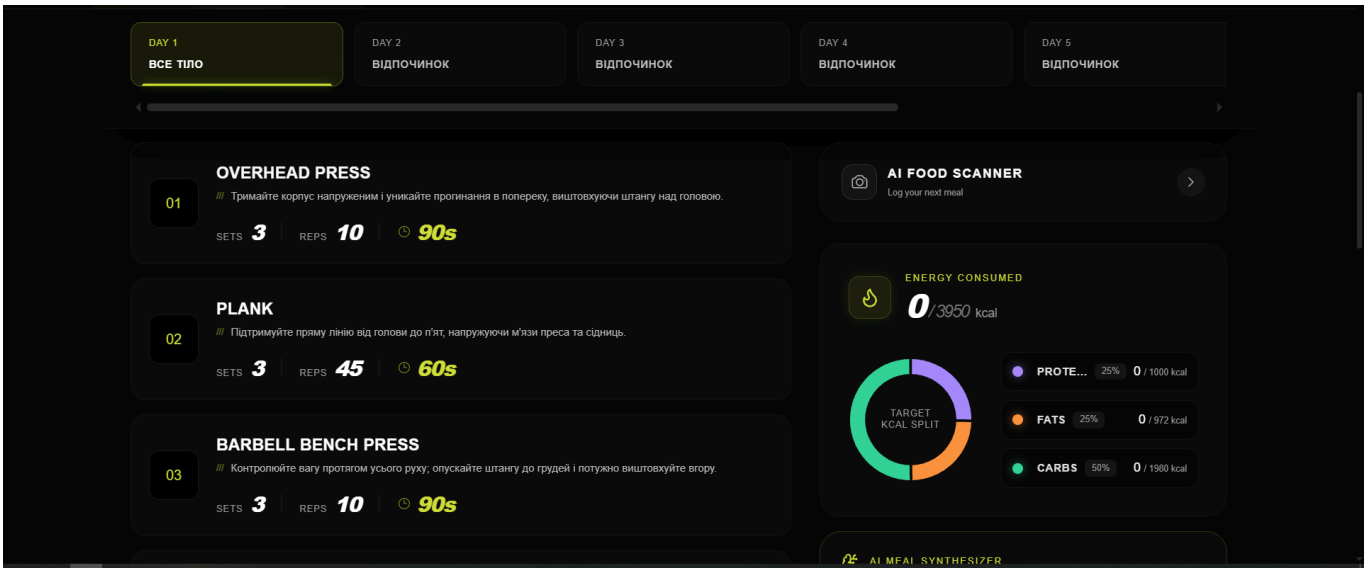

Рис. 3.14. Інтелектуальна панель моніторингу харчування

Джерело: розроблено автором

Права колонка (Nutrition Pane) відповідає за моніторинг харчування і є найтехнологічнішим візуальним елементом сторінки. Якщо для поточного дня ще не створено цілей харчування, система пропонує згенерувати їх в один клік (звернення до AI-генератора), після чого інтерфейс миттєво оновлюється. Основний акцент тут зроблено на інфографіку: кругова діаграма (Donut Chart) демонструє цільовий відсотковий розподіл калорій, а кільцеві індикатори прогресу внизу екрана в режимі реального часу відображають рівень споживання білків, жирів, вуглеводів та води відносно встановленої норми.

Крім аналітичної функції, ця панель інтегрує потужні інструменти взаємодії з комп'ютерним зором та генеративними моделями. Секція «AI Food Scanner» (компонент-акордеон) дозволяє користувачу увімкнути камеру або завантажити фотографію страви для автоматичного розпізнавання її харчової цінності. Центральним елементом виступає блок «AI Meal Synthesizer». Він пропонує користувачу набір швидких пресетів (наприклад, «Smart Fill», «Hit Protein», «Energy Boost»). При натисканні кнопки «Generate Meal» система аналізує залишок вільних макронутрієнтів користувача на поточний день і динамічно генерує персоналізований рецепт, який ідеально вписується у залишкові ліміти. Усі

збережені прийоми їжі формують хронологічний список (Logged Meals Sequence) у нижній частині панелі, забезпечуючи зручний аудит денного раціону.

Панель адміністрування системи (Admin Panel)

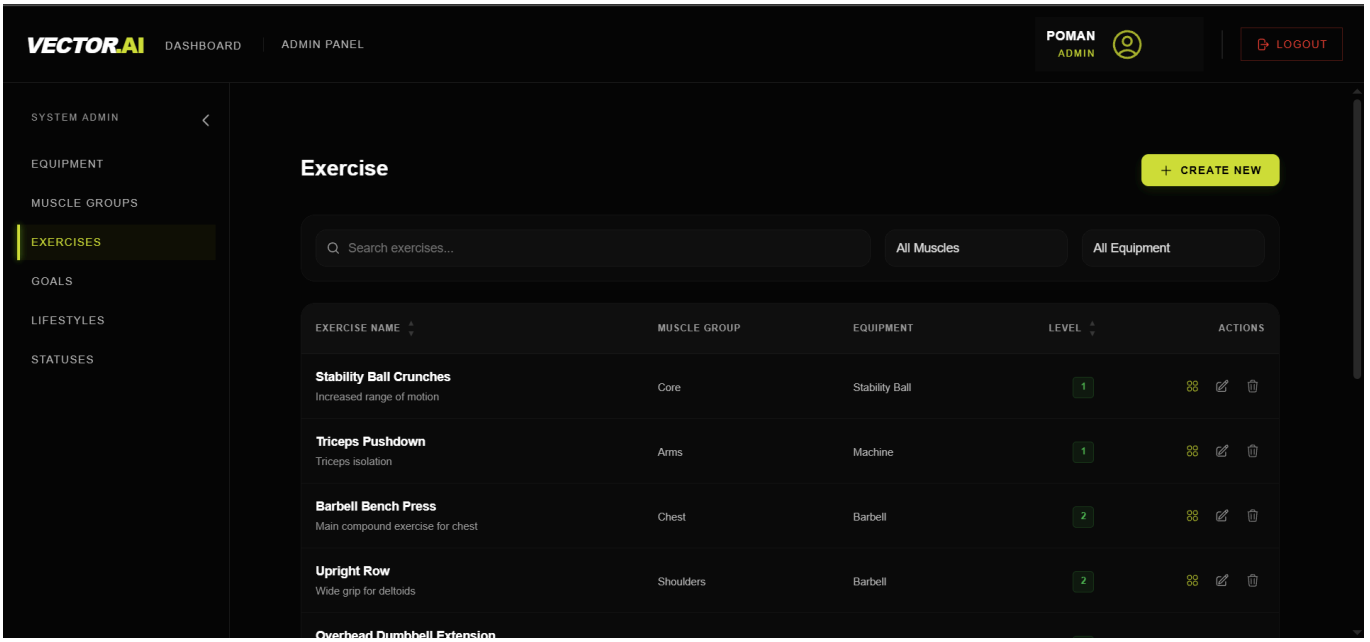

Рис. 3.15. Інтерфейс панелі адміністратора

Джерело: розроблено автором

Панель адміністрування (Admin Panel) є критично важливим закритим модулем системи Vector, доступ до якого регулюється суворою рольовою моделлю (Role-Based Access Control — RBAC). Лише користувачі з підтвердженим статусом адміністратора (перевірка здійснюється на етапі декодування JWT-токена) можуть потрапити до цього розділу. Головне призначення панелі — централізоване управління глобальними довідниками та налаштуваннями платформи. Оскільки якість згенерованих штучним інтелектом планів безпосередньо залежить від чистоти та структурованості вхідних даних, адміністративний інтерфейс спроектовано так, щоб забезпечити максимально зручний і безпомилковий процес наповнення бази.

Візуальна архітектура панелі побудована за класичним для складних систем патерном «Dashboard Layout». Ліворуч розташоване стаціонарне бокове навігаційне меню (Sidebar), яке забезпечує миттєвий доступ до управління всіма ключовими сутностями предметної області: обладнанням (Equipment), м'язовими групами

(Muscle Groups), вправами (Exercises), цілями (Goals), способами життя (Lifestyles) та статусами (Statuses). Верхня навігаційна панель (Header) відображає поточний контекст роботи, інформацію про профіль адміністратора та надає можливість безпечного завершення сесії (Logout).

Центральним елементом інтерфейсу на наведеному прикладі є інтерактивна таблиця даних (Data Grid), яка реалізує повний цикл CRUD-операцій (Create, Read, Update, Delete) для бази вправ. Зважаючи на те, що бібліотека вправ може налічувати сотні позицій, інтерфейс обладнано потужною системою пошуку та фільтрації. Верхній блок інструментів містить текстове поле для пошуку за назвою, а також випадаючі списки (Dropdowns) для реляційної фільтрації за м'язовою групою та типом обладнання. Технічно ці фільтри можуть працювати у комбінації, динамічно оновлюючи стан таблиці через локальний стан React або надсилаючи оптимізовані запити до серверного API з використанням техніки дебаунсингу (debouncing) для зниження навантаження на сервер.

Сама таблиця спроектована з акцентом на високу читабельність (Readability) та швидке сканування інформації. Кожен рядок не лише виводить сухі дані, але й використовує мікроформатування: під назвою вправи сірим кольором відображається короткий опис, що допомагає адміністратору швидше ідентифікувати елемент. Колонка рівня складності (Level) використовує колірне кодування у вигляді бейджів, що створює візуальну ієрархію даних. Заголовки таблиці оснащені індикаторами сортування, що дозволяє впорядковувати масиви даних за алфавітом або за зростанням/спаданням складності.

У правій частині кожного рядка розташований блок швидких дій (Actions), що складається з мінімалістичних іконок: перегляд деталей, редагування та видалення. Головна цільова дія сторінки — додавання нового елемента до бази — винесена у верхній правий кут у вигляді яскравої акцентної кнопки «CREATE NEW». Це зберігає консистентність загального дизайну платформи, де основні тригери створення завжди виділяються фірмовим лаймовим кольором на темному тлі.

Як і весь клієнтський застосунок, панель адміністрування розроблена з урахуванням принципів адаптивності. На мобільних пристроях чи екранах з невеликою роздільною здатністю бокове меню трансформується у прихований «гамбургер»-формат (Off-canvas menu), а широка таблиця набуває властивості горизонтального прокручування або трансформується у формат окремих карток, що дозволяє адміністратору оперативно вносити зміни до системи навіть зі смартфона.

Таким чином, розроблена клієнтська частина (Frontend) системи Vector являє собою сучасний, високопродуктивний та масштабований вебзастосунок. Завдяки використанню екосистеми React у поєднанні зі строгою типізацією TypeScript вдалося створити надійний інтерфейс, здатний безперебійно обробляти складні структури даних, що генеруються штучним інтелектом. Застосування архітектури Redux Toolkit забезпечило стабільне та передбачуване управління глобальним станом програми, зокрема процесами авторизації, моніторингом фонової генерації планів та збереженням користувацького контексту.

Критично важливим досягненням клієнтської розробки стала реалізація адаптивного та інтуїтивно зрозумілого інтерфейсу користувача (UI/UX). Використання фреймворку Tailwind CSS дозволило створити гнучку дизайн-систему за принципом Mobile-First, яка гарантує 100% збереження функціональності як на десктопних моніторах, так і на екранах мобільних пристроїв. Мінімізація візуального шуму, використання темної теми та акцент на мікроінтеракціях (анімації Framer Motion, інформативні сповіщення) суттєво знижують когнітивне навантаження на користувача.

Загалом, створений користувацький інтерфейс успішно виконує свою головну функцію — абстрагує користувача від складних внутрішніх алгоритмів бекенду та моделей штучного інтелекту, надаючи йому простий, зручний та естетично привабливий інструмент для досягнення персональних цілей у фітнесі та нутриціології.

3.5. Керівництво користувача

Цей розділ створено для того, щоб допомогти кінцевому споживачеві опанувати принципи роботи інтелектуальної платформи Vector та надати чіткі інструкції щодо експлуатації її базового інструментарію. Хоча інтерфейс застосунку розроблено з акцентом на інтуїтивність (Intuitive UI) задля максимального зниження когнітивного навантаження, користувачеві слід дотримуватися певної послідовності кроків, щоб розкрити весь потенціал алгоритмів штучного інтелекту.

Старт роботи та авторизація Перше знайомство із системою відбувається на головній сторінці (Home Page), яка презентує ключові можливості продукту. Щоб розпочати використання сервісу, необхідно натиснути цільову кнопку «Get Started», яка миттєво перенаправляє на форму ідентифікації. Платформа Vector відмовилася від класичної схеми реєстрації за допомогою створення логіна та пароля, що мінімізує небезпеку викрадення чи втрати облікових даних. Замість цього впроваджено сучасний протокол авторизації OAuth 2.0, який гарантує безпечний вхід через існуючий обліковий запис Google. Клієнту достатньо лише клікнути кнопку входу та вибрати свій Google-профіль. Щойно система успішно верифікує токен, вона створює активну сесію та переносить користувача до його особистого робочого середовища.

Внесення біометричних даних (Первинне налаштування) Коли клієнт вперше потрапляє до системи, вона автоматично фіксує відсутність антропометричних відомостей і тимчасово блокує функцію генерації планів, пропонуючи заповнити персональну анкету. Користувачу слід відкрити режим налаштувань («Setup Profile») і крок за кроком ввести свої базові показники: біологічну стать, поточну вагу, зріст та дату народження. Після цього необхідно обрати рівень своєї повсякденної активності (від малорухливого стилю життя до екстремальних навантажень), оцінити власний спортивний бекграунд, а також зазначити наявні специфічні дієтичні потреби (наприклад, вегетаріанство) та можливі харчові алергії. Грамотне та точне заповнення цієї форми є критично важливим, адже саме ці параметри виступають фундаментальним контекстом для

математичних моделей та ШІ-алгоритмів під час обчислення базового метаболізму та формування безпечного тренувального навантаження.

Генерація індивідуального фітнес-плану Маючи повністю заповнений профіль, користувач отримує доступ до конструювання власної тренувальної програми. Ця процедура реалізована у форматі покрокового інтерактивного майстра (Wizard) і включає такі етапи:

- **Визначення цілі та інтенсивності:** На старті необхідно обрати свій глобальний пріоритет (схуднення, гіпертрофія чи рекомпозиція тіла), встановити бажаний рівень складності (від 1 до 10) та, за необхідності, активувати режим занять виключно в домашніх умовах.
- **Налаштування розкладу:** Клієнт визначає комфортну кількість тренувальних днів на тиждень, бажану тривалість однієї сесії у хвилинах, загальну тривалість макроциклу (наприклад, 30 днів), а також обирає мову згенерованого штучним інтелектом контенту.
- **Фокус на м'язах:** Використовуючи зручний інтерактивний селектор, можна вказати цільові м'язи для акцентованого опрацювання або ж суворо виключити певні групи м'язів з плану (що особливо актуально у випадку наявних травм).
- **Відбір інвентарю:** Для тренувань у залі передбачено можливість відмітити доступне спортивне обладнання, щоб нейромережа не призначала вправи на тих тренажерах, яких немає в наявності.
- **Фінальна верифікація:** На завершальному кроці на екран виводиться зведене досьє з усіма обраними налаштуваннями. Кліком на кнопку «Initialize» користувач відправляє запит на обробку до AI-сервісу.

Зважаючи на те, що процес створення плану є ресурсомістким, система повертає користувача на головну панель (Dashboard), де транслюється актуальний статус виконання операції. Щойно процес генерації завершиться, програма стане доступною для активації.

Регулярна взаємодія: моніторинг тренувань та дієти Головним робочим простором для щоденної активності слугує інформаційна панель (Dashboard). Тут користувач може відстежувати власний прогрес та одним кліком відкривати сторінку

тренування, запланованого на поточний день. У режимі детального перегляду відображається перелік вправ із точними параметрами кількості підходів, повторень, часу на відпочинок, а також з персоналізованими порадами від AI.

Окремим щоденним завданням є контроль раціону харчування. Якщо добове меню ще не сформоване, клієнт може згенерувати його одним натисканням у панелі нутриціології. Для зручного логування спожитої їжі інтегровано функцію AI Food Scanner. Користувачу достатньо зробити фотографію своєї страви через смартфон або завантажити готове зображення з галереї. Нейромережа автоматично розпізнає страву, вираховує її калорійність і співвідношення макронутрієнтів (БЖВК), після чого самостійно додає ці дані до денного логу та оновлює індикатори прогресу. Якщо наприкінці доби залишається невикористаний ліміт калорій, можна скористатися інструментом AI Meal Synthesizer, щоб алгоритм підібрав ідеальний рецепт страви для закриття добової норми.

Панель адміністрування (тільки для авторизованого персоналу) Особи, яким присвоєно роль адміністратора системи, мають доступ до спеціальної закритої панелі управління (Admin Panel). Перехід до цього розділу здійснюється через відповідний пункт навігаційного меню. Інтерфейс адміністратора створено для централізованого управління глобальними довідниками платформи (списками вправ, обладнання, статусів та м'язових груп). Завдяки ергономічним таблицям із підтримкою пошуку та багаторівневої фільтрації адміністратор може легко створювати нові записи, редагувати існуючі (змінювати рівень складності чи прив'язку до обладнання) або видаляти неактуальні дані.

Висновок до розділу 3

У межах третього розділу цього кваліфікаційного дослідження успішно завершено ключову стадію конструювання програмного рішення, а саме — практичне програмування та запуск розумної платформи Vector. Спираючись на попередньо розроблені математичні моделі та вивчення предметного середовища,

вдалося спроєктувати розгалужену інфраструктуру новітнього вебсервісу, яка об'єднала захищений бекенд, динамічний фронтенд і компоненти нейромереж в єдине ціле.

Під час написання коду було аргументовано вибір та залучено найсучасніші технологічні інструменти. Зокрема, серверну інфраструктуру побудовано на базі новітнього середовища .NET 10, що працює у зв'язці з реляційною базою даних PostgreSQL та інструментарієм Entity Framework Core для об'єктно-реляційного відображення. Визначальним інженерним кроком виявилася імплементація підходів Domain-Driven Design, чистої архітектури (Clean Architecture) та шаблону CQRS із застосуванням функціоналу MediatR. Це дозволило досягти абсолютної ізоляції доменного ядра від інфраструктурних деталей, гарантуючи в такий спосіб відмовостійкість, захищеність та високий потенціал для горизонтального масштабування під час пікових навантажень.

Особливу наукову та практичну вагу має глибоке впровадження інструментів штучного інтелекту в логіку застосунку. Написані алгоритми конструювання промптів дають змогу нейромережам динамічно створювати максимально унікальні тренувальні макроцикли й дієтичні меню, спираючись на всебічний контекст: від антропометрії до наявного в клієнта спортивного приладдя та особистих цілей. Додатково платформу інтегровано з технологіями візуального розпізнавання (AI Food Scanner) для автоматичного підрахунку макросів за фотознімком їжі, а також підключено сервіс Meal Synthesizer, що генерує індивідуальні рецепти під актуальний залишок нутрієнтів. Задля збереження високої швидкодії пристроїв ці важкі для системи обчислення делеговано асинхронним фоновим обробникам (Background Workers).

Створення візуального інтерфейсу (Frontend) відбулося за парадигмою односторінкового вебзастосунку (SPA) на фундаменті React та TypeScript, з використанням Redux Toolkit для централізованого управління станом програми. Значний акцент був зроблений на проектуванні користувацького інтерфейсу за методологією Mobile-First. Застосування інструментарію Tailwind CSS разом із

Framer Motion забезпечило випуск стильного, інтуїтивно зрозумілого та повністю адаптивного дизайну.

У тексті докладно охарактеризовано розробку головних екранів: від стартової сторінки та системи захищеного входу через Google OAuth до багатокрокового майстра конструювання планів, основного дашборду та закритої адміністративної зони. Крім цього, виведено строгі критерії до програмного та апаратного забезпечення, виконання яких гарантує безвідмовну роботу платформи як на серверах, так і на гаджетах клієнтів. Сформована інструкція для користувачів яскраво ілюструє продуманість логіки шляху клієнта та загальну ергономічність цифрового продукту. Окремо висвітлено вектори майбутньої модернізації сервісу, де доведено стратегічну необхідність розробки кросплатформного мобільного додатка на технології React Native, що дозволить глибше взаємодіяти із сенсорами смартфонів.

Зважаючи на підсумки третього розділу, інтелектуальний сервіс Vector постає як абсолютно працездатний, інноваційний та логічно завершений програмний комплекс. Обрані для проєкту архітектурні шаблони та успішна інтеграція AI-моделей підтверджують високий рівень інженерної реалізації, а отже, мета та завдання кваліфікаційної праці виконані в повному обсязі.

ВИСНОВОК

Підсумовуючи результати кваліфікаційного дослідження, варто зазначити, що в його межах було успішно подолано весь цикл інженерної розробки — починаючи з формування теоретичної концепції та математичного підґрунтя і закінчуючи випуском повноцінного full-stack продукту на базі хмарної інфраструктури AWS. Розроблена платформа Vector яскраво ілюструє не тільки потужність новітніх IT-фреймворків, але й критичну значущість нейромереж для глибокої індивідуалізації процесів тілесного вдосконалення та контролю харчування.

Архітектурна база та використані технології. Грамотно підібраний набір інструментів відіграв ключову роль у безпроблемному створенні як серверного, так і клієнтського боку системи. Зокрема, новітнє середовище .NET 10 довело свою беззаперечну надійність та стабільність під час конструювання бекенду. Визначальним фактором успіху стало безумовне слідування парадигмам предметно-орієнтованого програмування (DDD) спільно з підходами чистої архітектури (Clean Architecture). Інтеграція інструментарію MediatR для реалізації шаблону CQRS гарантувала високу адаптивність програмного комплексу, надійно ізолювавши процеси зміни стану від важких операцій зчитування даних, що виявилося фундаментальною умовою для підтримки швидкодії вебсервісу.

Інноваційне використання ШІ. У ході реалізації проєкту було впроваджено революційну методику конструювання спортивних циклів. Завдяки уникненню стандартних, заздалегідь прописаних схем і переходу до автоматизованої алгоритмічної генерації, вдалося спроєктувати платформу з максимальним рівнем індивідуалізації під кожного юзера. Написання глибоких контекстних запитів (промпт-інжиніринг) дозволило системі продукувати тренувальні графіки, що бездоганно корелюють із біометрією, наявним інвентарем та кінцевими амбіціями клієнта. Надзвичайно високу практичну цінність має залучення технологій комп'ютерного зору (AI Food Scanner) для візуального оцінювання страв, а також інтеграція генератора індивідуальних рецептів (Meal Synthesizer), які спільно трансформують контроль за раціоном у повністю автоматизований процес.

Математичне підґрунтя та бази даних. Побудова розумної системи була б неможливою без ретельного дослідження біологічних параметрів людського тіла. Структурування алгоритмів обчислення базового обміну речовин та калькуляції добового макросу (БЖВК) створило міцну базу для коректного функціонування нейромереж. До того ж розробка ретельно нормалізованої моделі бази даних на платформі PostgreSQL забезпечила строгу узгодженість збереженої інформації — починаючи від структури спортивних сесій і закінчуючи фіксацією щоденного раціону.

UI/UX дизайн та взаємодія. Вибір формату односторінкового застосунку (SPA) із застосуванням React 18 і строгої типізації TypeScript дозволив збудувати для користувача інтуїтивно зрозуміле та безперервне цифрове середовище. Орієнтація на концепцію Mobile-First та використання утиліт Tailwind CSS гарантували бездоганну кросплатформність і коректне відображення на мобільних телефонах, що виступає критичною вимогою для спортивних продуктів. Запровадження інструментарію Redux Toolkit вирішило виклики, пов'язані з глобальним станом додатка, тоді як грамотна реалізація анімацій та інтерфейсних мікрореакцій суттєво зменшила когнітивний тиск на юзера під час аналізу комплексних звітів.

Кіберзахист та оптимізація. Підсумки роботи беззаперечно доводять високу необхідність застосування передових практик для захисту приватності та пришвидшення фонових обчислень. Впровадження стандарту Google Auth (OAuth 2.0) у синергії з криптографічними токенами JWT забезпечило максимальний рівень недоторканності особистих відомостей. Разом з тим, делегування важких операцій (як-от обробка запитів до ШІ) в ізольовані фонові потоки свідчить про глибокий інженерний підхід, адже це вберегло головний потік від зависань і кардинально підвищило чуйність платформи.

Практичний потенціал та майбутній розвиток. Програмний комплекс, створений у межах роботи, перевершує формат звичайного університетського проєкту. Платформа Vector є потужним і самодостатнім цифровим помічником, здатним ефективно боротися з гіподинамією та проблемами харчування, притаманними сучасному стилю життя. Фіналізація поточного етапу розробки

прокладає шлях до подальшого масштабування продукту: отримані практичні навички створюють ідеальний фундамент для розробки нативних мобільних застосунків (через React Native), що відкриє доступ до фізичних сенсорів гаджетів та забезпечить автоматизований трекінг рухливості.

Отже, спираючись на вищезазначені факти, є всі підстави стверджувати про абсолютне та стовідсоткове виконання всіх завдань, сформульованих на початку дослідження. У процесі розробки автор продемонстрував впевнене володіння передовими інструментами програмування, а також уміння автономно розв'язувати нетривіальні завдання щодо проєктування архітектури та написання алгоритмів. Підсумковий продукт становить собою логічно завершене комплексне дослідження, що має високу комерційну привабливість і повністю готове до запуску та експлуатації на ринку сучасних Health & Fitness сервісів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ASP.NET Core Documentation. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core
2. Entity Framework Core Documentation. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/ef/core
3. PostgreSQL Documentation. [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs
4. MediatR Documentation. [Електронний ресурс]. – Режим доступу: https://github.com/jbogard/MediatR
5. FluentValidation Documentation. [Електронний ресурс]. – Режим доступу: https://docs.fluentvalidation.net
6. Cloudflare R2 Object Storage Documentation. [Електронний ресурс]. – Режим доступу: https://developers.cloudflare.com/r2
7. Google Gemini API Documentation. [Електронний ресурс]. – Режим доступу: https://ai.google.dev/docs
8. React Documentation. [Електронний ресурс]. – Режим доступу: https://react.dev
9. TypeScript Documentation. [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/docs
10. Vite Documentation. [Електронний ресурс]. – Режим доступу: https://vitejs.dev/guide
11. Tailwind CSS Documentation. [Електронний ресурс]. – Режим доступу: https://tailwindcss.com/docs
12. Framer Motion Documentation. [Електронний ресурс]. – Режим доступу: https://www.framer.com/motion
13. Redux Toolkit Documentation. [Електронний ресурс]. – Режим доступу: https://redux-toolkit.js.org
14. React Router Documentation. [Електронний ресурс]. – Режим доступу: https://reactrouter.com
15. Axios Documentation. [Електронний ресурс]. – Режим доступу: https://axios-http.com/docs/intro
16. Google OAuth 2.0 Documentation. [Електронний ресурс]. – Режим доступу: https://developers.google.com/identity/protocols/oauth2
17. Microsoft Azure Architecture Center. CQRS pattern. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs
18. Microsoft Learn. RESTful web API design. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

19. Docker Documentation. [Електронний ресурс]. – Режим доступу: https://docs.docker.com
20. Amazon Web Services (AWS) Documentation. [Електронний ресурс]. – Режим доступу: https://docs.aws.amazon.com
21. JSON Web Tokens (JWT) Introduction. [Електронний ресурс]. – Режим доступу: https://jwt.io/introduction
22. The Twelve-Factor App Methodology. [Електронний ресурс]. – Режим доступу: https://12factor.net
23. World Health Organization (WHO). Obesity and overweight. [Електронний ресурс]. – Режим доступу: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight
24. World Health Organization (WHO). Physical activity. [Електронний ресурс]. – Режим доступу: https://www.who.int/news-room/fact-sheets/detail/physical-activity
25. OpenAI API Documentation. [Електронний ресурс]. – Режим доступу: https://platform.openai.com/docs
26. Prompt Engineering Guide. [Електронний ресурс]. – Режим доступу: https://www.promptingguide.ai
27. OWASP Top 10:2021. The Ten Most Critical Web Application Security Risks. [Електронний ресурс]. – Режим доступу: https://owasp.org/Top10
28. MDN Web Docs: CSS Grid Layout. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout
29. MDN Web Docs: CSS Flexible Box Layout (Flexbox). [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout
30. Feature-Sliced Design (FSD) Architectural Methodology. [Електронний ресурс]. – Режим доступу: https://feature-sliced.design
31. xUnit.net Documentation. [Електронний ресурс]. – Режим доступу: https://xunit.net
32. Git Documentation. [Електронний ресурс]. – Режим доступу: https://git-scm.com/doc
33. Scalar API Reference. [Електронний ресурс]. – Режим доступу: https://scalar.com
34. Npgsql Entity Framework Core Provider Documentation (офіційна документація провайдера для роботи PostgreSQL з EF Core). [Електронний ресурс]. – Режим доступу: https://www.npgsql.org/efcore/
35. Material Tailwind Documentation (документація бібліотеки UI-компонентів, яку ти згадував у тексті). [Електронний ресурс]. – Режим доступу: https://www.material-tailwind.com/docs/react/installation

36. react-i18next Documentation (бібліотека для реалізації багатомовності інтерфейсу). [Електронний ресурс]. – Режим доступу: https://react.i18next.com/
37. Google Cloud Vision API Documentation (документація для реалізації AI Food Scanner та розпізнавання зображень). [Електронний ресурс]. – Режим доступу: https://cloud.google.com/vision/docs
38. Microsoft Learn. Background tasks with hosted services in ASP.NET Core (документація щодо створення фонових процесів Worker для генерації планів штучним інтелектом). [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services
39. Fowler M. CQRS (оригінальна стаття Мартіна Фаулера з детальним розбором патерну CQRS). [Електронний ресурс]. – Режим доступу: https://martinfowler.com/bliki/CQRS.html
40. Web Vitals. Essential metrics for a healthy site (документація від Google щодо метрик продуктивності вебзастосунків). [Електронний ресурс]. – Режим доступу: https://web.dev/articles/vitals
41. RFC 7807 – Problem Details for HTTP APIs (міжнародний стандарт обробки та стандартизації помилок у REST API, використаний у твоєму Exception Middleware). [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7807
42. IETF RFC 6749. The OAuth 2.0 Authorization Framework (офіційна специфікація протоколу авторизації OAuth 2.0). [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6749
43. U.S. Department of Agriculture (USDA). FoodData Central API (відкрите API з базами даних про нутрієнти та БЖВК продуктів, актуальне для нутриціологічної частини роботи). [Електронний ресурс]. – Режим доступу: https://fdc.nal.usda.gov/api-guide.html