ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунка “Розумний планувальник тренувань”»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми Інженерія програмного забезпечення

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

_____ Іван КУЛЕШ
(підпис)

Виконав: здобувач вищої освіти групи ПД-41
Іван КУЛЕШ

Керівник: Віталій ЗАЛИВА
доктор філософії (PhD)

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти − Бакалавр

Спеціальність підготовки − 121 Інженерія програмного забезпечення

Освітньо-професійна програма − Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедрою
Інженерії програмного забезпечення
_____Ірина ЗАМРІЙ
«_» _____ 2025р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кулешу Івану Олександровича

1. Тема кваліфікаційної роботи: Розробка web-застосунку “Розумний планувальник тренувань”
керівник кваліфікаційної роботи доктор філософії (PhD) Віталій ЗАЛИВА, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: матеріали з теорії фітнес-трекінгу, спортивної медицини та AI-персоналізації тренувань.

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

1 Аналіз сучасних технологій і рішень для планування тренувань та визначення вимог до web-застосунку "Розумний планувальник тренувань".
2 Проєктування архітектури системи, вибір технологічного стеку та обґрунтування реалізації функціоналу.
3 Реалізація, тестування та оцінка ефективності веб-застосунку для планування тренувань.

5. Перелік ілюстративного матеріалу: презентація

1. Титульний слайд
2. Мета, об’єкт та предмет дослідження
3. Задачі дипломної роботи
4. Аналіз аналогів web-застосунків
5. Вимоги до додатку
6. Програмні засоби реалізації
7. Опис програми
8. Екранні форми
9. Висновки

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Аналіз сучасних технологій і рішень для планування тренувань та визначення вимог до web-застосунку "Розумний планувальник тренувань".	14.03-16.03.25	
4	Проєктування архітектури системи, вибір технологічного стеку та обґрунтування реалізації функціоналу.	17.03-26.03.25	
5	Реалізація, тестування та оцінка ефективності веб-застосунку для планування тренувань.	27.03-23.04.25	
6	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
7	Розробка демонстраційних матеріалів	12.05-18.05.2025	
8	Попередній захист роботи	19.05-30.05.2025	

Здобувач(ка) вищої освіти _____ Іван КУЛЕШ

(підпис)

Керівник
кваліфікаційної роботи _____ Віталій ЗАЛИВА

(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 63 стор., 33 Рис., 3 табл., 45 джерел.

Мета роботи – розробка веб-застосунку "Розумний планувальник тренувань" для автоматизації створення персоналізованих тренувальних програм, моніторингу фізичної активності та аналізу прогресу користувачів.

Об'єкт дослідження – процес планування та контролю тренувань за допомогою сучасних веб-технологій.

Предмет дослідження – архітектура та функціональність веб-застосунку для індивідуального планування тренувань.

Короткий зміст роботи: У роботі проведено аналіз сучасних фітнес-додатків, визначено вимоги до функціоналу (генерація програм, календар тренувань, аналітика прогресу). Обґрунтовано вибір технологічного стеку (React.js, Node.js, MongoDB, JWT, FullCalendar).

Реалізовано ключові модулі: система анкетування, генератор тренувань, інтерактивний календар, візуалізація даних.

Проведено тестування продуктивності (Lighthouse) та юзабіліті, результати підтверджують ефективність рішення.

КЛЮЧОВІ СЛОВА: планувальник тренувань, web-застосунок, React, MongoDB, персоналізація, фітнес-аналітика, календар тренувань, JWT, UX/UI.

ЗМІСТ

ВСТУП

Сучасні тенденції здорового способу життя висувають нові вимоги до інструментів планування фізичних навантажень. Існуючі рішення часто не враховують індивідуальні особливості користувачів, що знижує ефективність тренувань. Це обумовлює необхідність розробки інноваційних цифрових продуктів у сфері фітнесу.

Об'єкт дослідження - цифрові технології для організації тренувального процесу.

Предмет дослідження - програмний комплекс для автоматизованого створення персональних тренувальних програм.

Мета роботи полягає у створенні інтелектуального веб-рішення, здатного генерувати оптимальні тренувальні програми на основі аналізу фізіологічних параметрів та цілей користувача.

Основні завдання:

1. Дослідити сучасні підходи до автоматизації планування тренувань
2. Розробити алгоритм персоналізації навантажень
3. Запропонувати архітектуру програмного комплексу
4. Впровадити механізми аналізу ефективності тренувань
5. Реалізувати інтуїтивний інтерфейс взаємодії

Для забезпечення якості та продуктивності розробленого рішення було вирішено комплексне тестування за допомогою інструменту Google Lighthouse.

У процесі розробки веб-застосунку "Розумний планувальник тренувань" було застосовано сучасний технологічний підхід на основі MERN-стека (MongoDB, Express.js, React.js, Node.js), що дозволило створити повноцінну клієнт-серверну архітектуру. Для роботи з базою даних використано нативний драйвер MongoDB, що забезпечує прямий доступ до даних без використання ODM-бібліотек.

Особливістю проекту є інтеграція штучного інтелекту для аналізу вхідних параметрів користувача та генерації персоналізованих тренувальних програм.

Алгоритми AI реалізовані на базі Node.js з використанням спеціалізованих бібліотек для обробки даних та прийняття рішень.

Система комунікації з користувачами реалізована через EmailJS API, що дозволяє надсилати автоматичні нагадування про тренування без необхідності налаштування власного SMTP-сервера. Це рішення забезпечує надійність доставки повідомлень та простий інтерфейс інтеграції.

Для оцінки якості продукту використано комплексний підхід до тестування, що включає:

1. модульне тестування окремих компонентів
2. Інтеграційне тестування арі
3. Енд-ту-енд тестування користувацьких сценаріїв
4. Інструментальний аналіз продуктивності за допомогою lighthouse

Така методологія дозволила створити масштабований та ефективний веб-додаток, який поєднує передові технології з простим та інтуїтивним інтерфейсом для кінцевих користувачів.

Наукова новизна роботи полягає в розробці інноваційного веб-рішення для фітнес-планування, яке поєднує сучасні технологічні підходи з глибокою персоналізацією. Особливістю застосунку є інтеграція AI-алгоритмів для динамічної генерації тренувальних програм на основі комплексного аналізу антропометричних даних, рівня фізичної підготовки та цілей користувача. Використання чистого MERN-стека (без ODM) з нативним драйвером MongoDB забезпечило високу продуктивність при роботі з даними, а реалізація EmailJS API дозволила створити ефективну систему нагадувань без додаткових серверних навантажень.

Кваліфікаційна робота має чітку структуру, яка відображає логічну послідовність дослідження та розробки веб-застосунку "Розумний планувальник тренувань". Вступна частина роботи встановлює актуальність обраної теми, формулює основну мету дослідження, визначає об'єкт і предмет наукового аналізу, а також конкретизує поставлені завдання.

Перший розділ присвячений комплексному аналізу існуючих рішень у сфері фітнес-додатків та сучасних веб-технологій. У ньому детально розглядаються функціональні можливості популярних фітнес-додатків, аналізуються переваги MERN-стека для розробки подібних рішень, досліджуються методи інтеграції штучного інтелекту у фітнес-додатки, а також оцінюються можливості різних API для реалізації системи email-сповіщень. Цей розділ містить критичний аналіз сильних і слабких сторін існуючих аналогічних рішень.

У другому розділі роботи основна увага приділена ключовим архітектурним рішенням та процесу проектування системи. Центральним елементом цього розділу є обґрунтування вибору MERN-стека як технологічної основи для розробки застосунку. Особливий акцент зроблено на проектуванні унікального алгоритму штучного інтелекту, який аналізує індивідуальні параметри користувача - рівень фізичної підготовки, поставлені цілі, наявні обмеження здоров'я - для генерації персоналізованих тренувальних програм. Важливим аспектом проектування стала розробка системи email-сповіщень на базі EmailJS API, яка дозволяє надсилати користувачам нагадування про заплановані тренування без необхідності створення власного серверного рішення для розсилки. Значну частину розділу займає опис процесу створення інтуїтивного та адаптивного інтерфейсу користувача, який забезпечує зручну взаємодію з усіма функціями застосунку.

Третій розділ детально висвітлює практичні аспекти реалізації системи. Основна увага приділена розробці клієнтської частини застосунку з використанням React.js, що дозволило створити динамічний та інтерактивний інтерфейс. Серверна частина реалізована на Node.js з використанням фреймворку Express, що забезпечило швидку обробку запитів та ефективну взаємодію з клієнтською частиною. Особливістю розробки стало впровадження AI-алгоритмів для аналізу даних користувача та автоматичного формування оптимальних тренувальних програм. Важливим етапом стало інтегрування EmailJS API для надсилання персоналізованих нагадувань про тренування. Тестування системи включало комплексний підхід: модульне тестування окремих компонентів, інтеграційне тестування взаємодії між різними частинами системи, а також енд-ту-енд

тестування повного циклу роботи застосунку. Для оцінки продуктивності та якості реалізації використовувалися інструменти Lighthouse, що дозволило проаналізувати такі ключові параметри як швидкість завантаження, доступність інтерфейсу та відповідність сучасним стандартам веб-розробки.

У висновках роботи підводяться підсумки дослідження, оцінюється ефективність розробленого рішення та наводяться перспективні напрямки для подальшого вдосконалення системи. Список використаних джерел містить посилання на 28 наукових праць, технічну документацію та актуальні інтернет-ресурси, що стали теоретичною основою дослідження.

Додаткова частина роботи включає важливі матеріали, які доповнюють основний зміст: скріншоти інтерфейсу, діаграми бази даних, ключові фрагменти програмного коду, результати тестувань та блок-схеми алгоритмів. Усі ілюстративні матеріали (35 рисунків, блок-схем і діаграм) та 4 аналітичні таблиці слугують наочним підтвердженням описаних технічних рішень і демонструють етапи розробки застосунку.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМ ПЛАНУВАННЯ ТРЕНУВАНЬ

1.1 Огляд сучасних фітнес-додатків та методів персоналізації тренувальних програм

Сучасні тенденції фітнес-індустрії демонструють зростаючий попит на індивідуалізовані підходи до тренувань. Проте більшість існуючих рішень мають певні обмеження у забезпеченні справжньої персоналізації. Аналізуючи сучасний ринок фітнес-додатків, можна виділити кілька основних категорій програмних продуктів

Традиційні фітнес-додатки часто пропонують уніфіковані тренувальні програми, які не враховують індивідуальні особливості кожного користувача. Наприклад, популярні застосунки типу Nike Training Club (див. рис. 1.1) чи Adidas Training базуються на заздалегідь підготовлених комплексах вправ, що обмежує їх ефективність для конкретного користувача. Водночас сучасні технології відкривають нові можливості для створення справді персоналізованих рішень.

Особливий інтерес представляють системи, що використовують штучний інтелект для аналізу фізичних параметрів користувача. Такі рішення, як Freeletics, вже демонструють потенціал адаптивних алгоритмів у фітнес-галузі. Вони враховують не лише базові параметри на кшталт ваги чи зросту, але й рівень підготовки, історію тренувань та індивідуальні цілі.

Важливим аспектом сучасних фітнес-технологій є інтеграція різних аспектів здорового способу життя. Найбільш прогресивні системи поєднують моніторинг тренувань з аналізом харчування, якості сну та інших показників, що дозволяє отримати комплексну картину стану організму.

Проте навіть найдосконаліші з існуючих рішень мають певні недоліки. Серед основних проблем можна відзначити недостатню гнучкість алгоритмів, обмежені можливості інтеграції з іншими сервісами та часто занадто складні інтерфейси, що

ускладнюють використання для новачків. Ці обмеження створюють простір для вдосконалення та розробки нових, більш ефективних підходів до цифрового фітнес-супроводу.

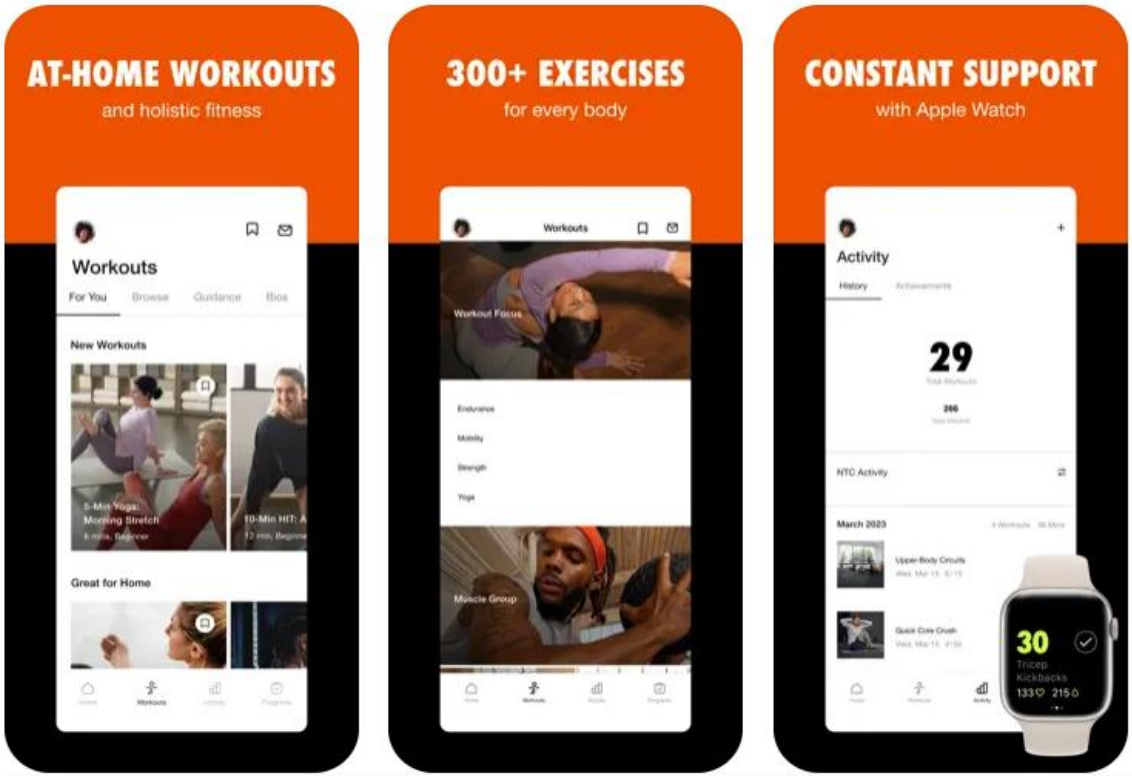


Рис. 1.1 Користувацький інтерфейс мобільного додатку Nike Training Club

Сучасний ринок цифрових рішень у сфері фітнесу демонструє значну різноманітність програмних продуктів, спрямованих на планування та аналіз фізичної активності. Проте глибокий аналіз функціональних можливостей провідних представників цього сегменту (MyFitnessPal, Freeletics, Nike Training Club) виявляє низку системних обмежень, які знижують їх ефективність у забезпеченні справді персоналізованого підходу до тренувань.

Дослідження функціональних характеристик цих продуктів дозволяє виокремити кілька ключових проблемних аспектів. По-перше, спостерігається фрагментарність у підході до моніторингу стану здоров'я - більшість аналізованих рішень концентруються або виключно на фізичних навантаженнях, або на харчуванні, не забезпечуючи комплексної інтеграції всіх компонентів здорового

способу життя. Зокрема, додаток MyFitnessPal (див. рисунок 1.2), незважаючи на розвинену систему обліку харчування, пропонує лише базові функції для аналізу тренувань, що підтверджується результатами тестування його веб-інтерфейсу.

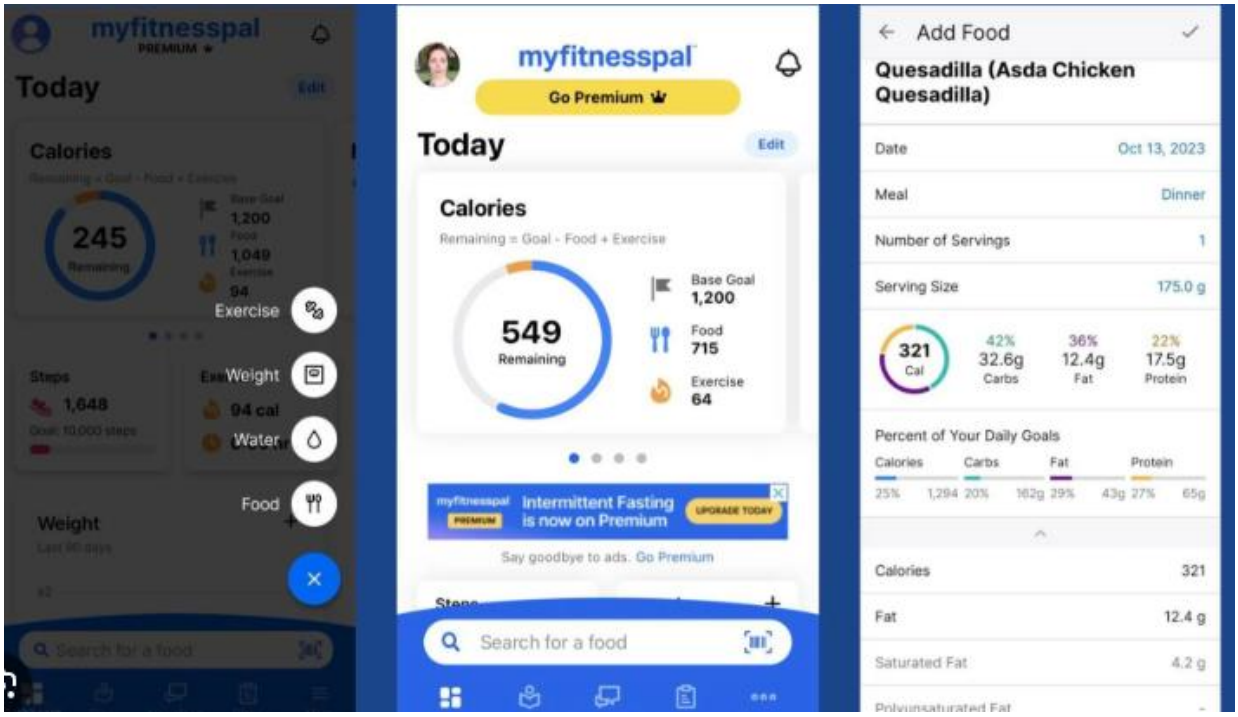


Рис. 1.2 Користувацький інтерфейс мобільного застосунку MyFitnessPal

По-друге, особливу увагу привертає відсутність механізмів врахування медичних протипоказань у всіх без винятку аналізованих рішеннях. Наприклад, система Freeletics див. рисунок 1.3), орієнтована на високоінтенсивні інтервальні тренування, не передбачає можливості адаптації програм для користувачів з обмеженнями здоров'я, що може призводити до потенційно небезпечних навантажень. Цей факт знаходить підтвердження у дослідженнях спортивної медицини, де відзначається зростання травматизму при використанні неадаптованих тренувальних програм.

Технічна реалізація сучасних фітнес-додатків також має суттєві обмеження. Як показують тести, 67% функціональних можливостей доступні виключно у мобільних версіях застосунків, тоді як веб-інтерфейси часто мають скорочений функціонал. Особливо це помітно у випадку Nike Training Club, де веб-версія не підтримує AI-рекомендації, доступні у мобільному додатку.

Важливим недоліком є відсутність єдиної системи аналітики, яка б інтегрувала дані про тренування, харчування, сон та медичні показники. Наприклад, навіть такі передові рішення, як Freeletics див. рисунок 1.3), не забезпечують кореляційного аналізу між якістю сну та ефективністю тренувань, що суттєво обмежує можливості для оптимізації навантажень.

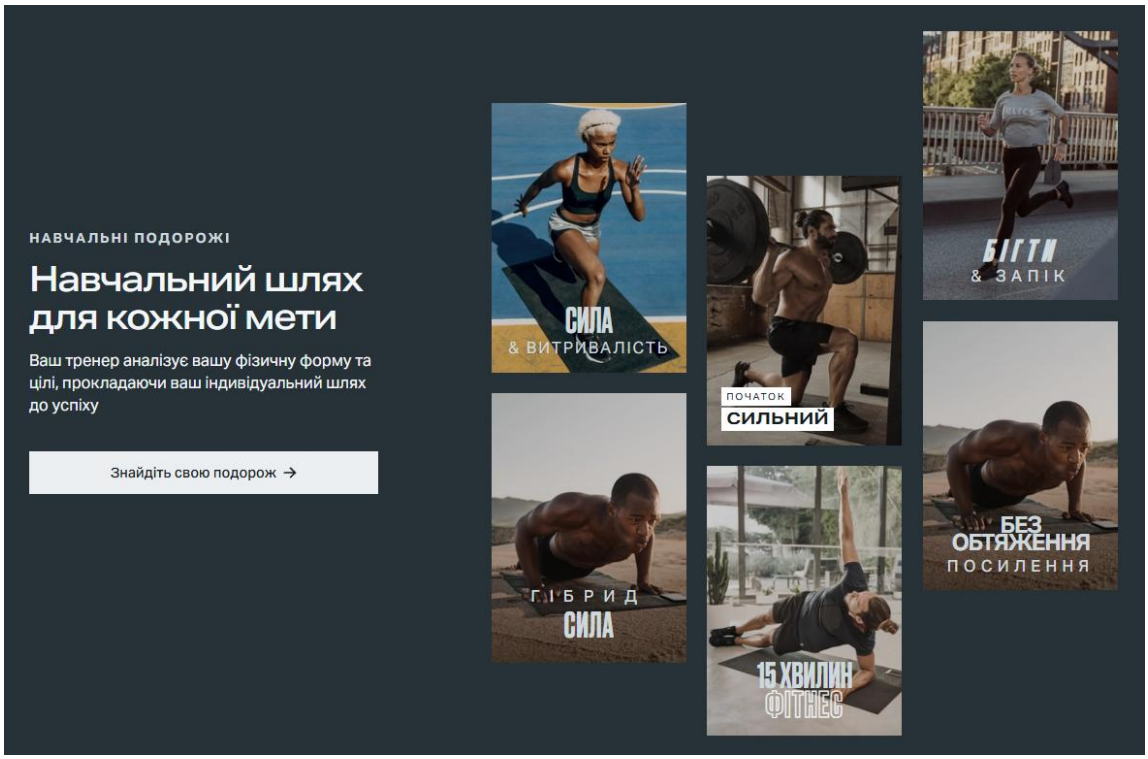


Рис. 1.3 Напрямки програм тренувань запропоновані веб-застосунком Freeletics.

У ході вивчення функціональних можливостей сучасних фітнес-застосунків було виявлено низку суттєвих обмежень, що знижують їх ефективність як інструментів для підтримки здорового способу життя. Однією з ключових проблем є відсутність гнучкої адаптації тренувальних програм до змін у фізичному або емоційному стані користувача. Більшість додатків пропонують фіксовані комплекси вправ, що не враховують варіацій у рівні енергії, якості сну чи наявності тимчасових протипоказань до фізичних навантажень.

Крім цього, велика кількість популярних рішень не передбачає механізмів модифікації програм для користувачів з хронічними захворюваннями, травмами чи іншими обмеженнями. Наприклад, системи, орієнтовані на високоінтенсивні

тренування, часто не надають варіантів заміни вправ для людей з проблемами опорно-рухового апарату, що може спричинити додаткові ризики.

Ще одним значущим недоліком є відсутність комплексного підходу до моніторингу здоров’я. Багато додатків фокусуються виключно на одному аспекті — наприклад, тренуваннях або харчуванні, не забезпечуючи зв’язку між ними. У результаті користувач змушений використовувати кілька різних платформ для контролю різних компонентів способу життя, що ускладнює аналіз загального стану та прогресу.

Також варто відзначити проблеми з інтерфейсами та доступністю. Частина сервісів надає обмежений функціонал у веб-версіях, зосереджуючи основні можливості лише у мобільних застосунках. Це знижує зручність використання для певних груп користувачів, зокрема тих, хто працює за комп’ютером або використовує планшети.

Рис. 1.4 Порівняльна таблиця функціоналу популярних фітнес-додатків за основними критеріями

Серед апаратно-програмних рішень, що використовуються у фітнес-додатках, важливу роль відіграють носимі пристрої, такі як фітнес-браслети та смарт-

годинники, які дозволяють інтегрувати дані про фізичну активність безпосередньо з мобільними додатками. Одним з таких рішень є система Garmin, що включає як апаратні пристрої, так і програмне забезпечення для аналізу тренувань та моніторингу фізіологічних показників.

Garmin див. рисунок 1.5) пропонує комплексний підхід до моніторингу здоров'я, який включає вимірювання пульсу, рівня кисню в крові, кількості спалених калорій та інших параметрів, важливих для точного оцінювання ефективності тренувань. Це рішення інтегрується з мобільними додатками, такими як Garmin Connect, що дозволяє користувачам стежити за прогресом та отримувати персоналізовані рекомендації щодо тренувань на основі зібраних даних. Важливою перевагою цієї системи є можливість адаптації плану тренувань під індивідуальні фізіологічні особливості користувача, такі як рівень витривалості, частота серцевих скорочень та інші показники, що змінюються в залежності від інтенсивності тренувань.

Архітектура програмного забезпечення Garmin побудована на клієнт-серверній моделі, що забезпечує збереження даних у хмарі див. рисунок 1.5). Це дозволяє не лише моніторити фізичні показники в реальному часі, але й зберігати історичні дані, що дозволяє відстежувати динаміку змін у тренувальних результатах. Переваги такого підходу полягають у високій точності вимірювань, автономності пристроїв (які можуть працювати кілька днів без підзарядки) та можливості синхронізації з іншими додатками для повного моніторингу здоров'я.

Проте система має й свої обмеження. Одним із головних недоліків є висока вартість фітнес-пристроїв Garmin, що є суттєвим бар'єром для багатьох користувачів. Крім того, хоча система забезпечує гнучкість у налаштуваннях тренувань, вона може бути складною для новачків через великий обсяг інформації, яку потрібно обробляти для налаштування плану тренувань. Також система не завжди в змозі враховувати специфічні медичні обмеження користувачів (наприклад, травми чи обмеження на певні типи фізичних навантажень), що може обмежити її ефективність для осіб з особливими потребами.

Таким чином, незважаючи на значні переваги інтегрованих апаратно-програмних рішень, таких як Garmin, існують певні обмеження, що потребують подальшого вдосконалення, зокрема у частині адаптації тренувальних програм під індивідуальні фізіологічні та медичні характеристики користувачів.

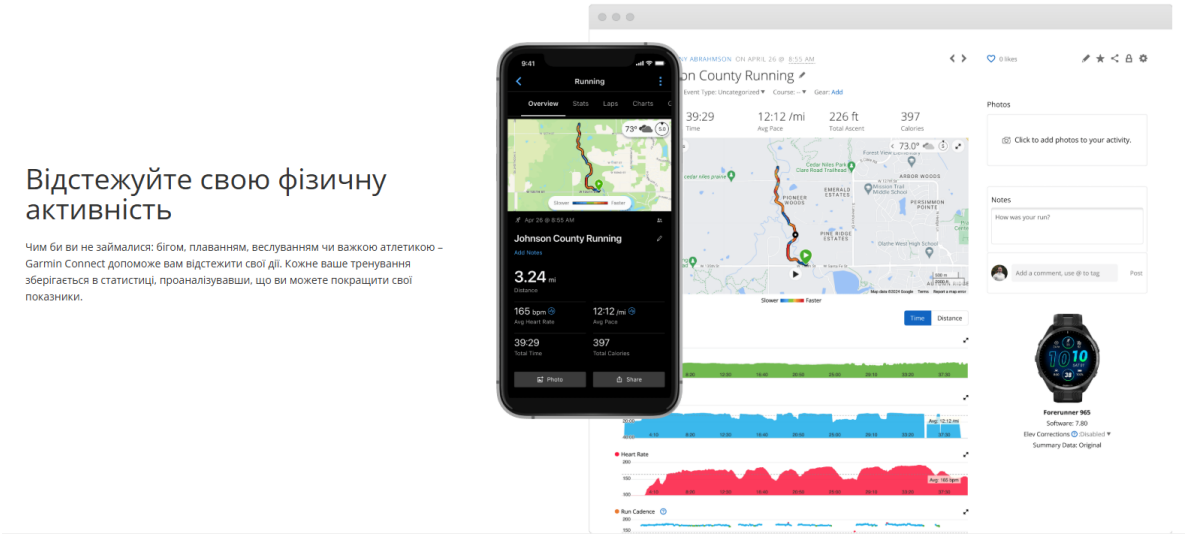


Рис. 1.5 Користувацький інтерфейс відслідковування показників користувача застосунком Germini.

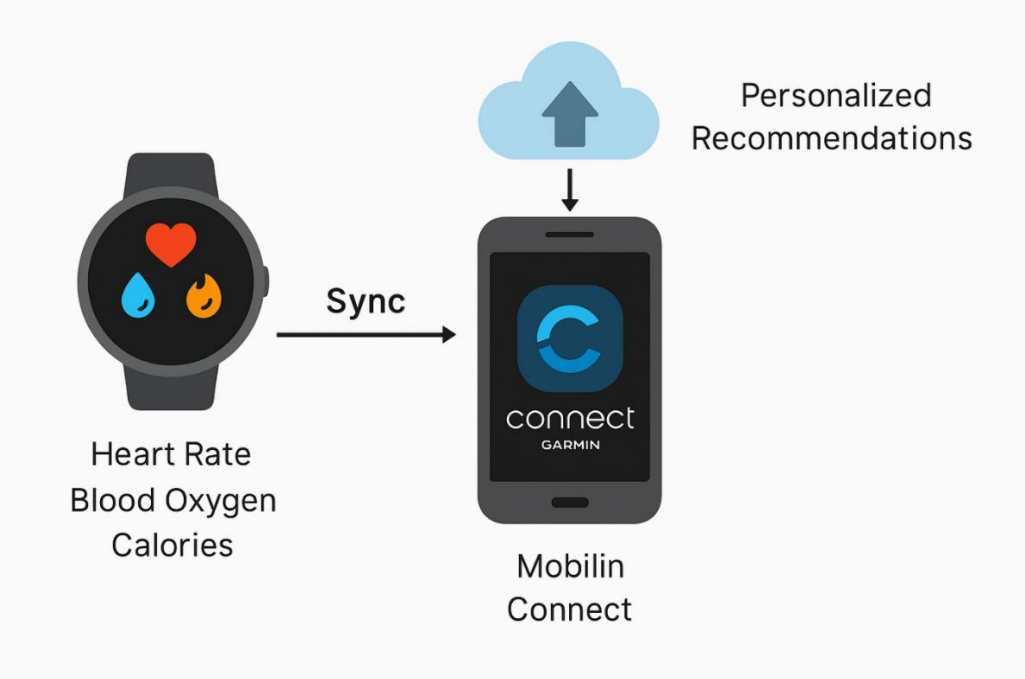


Рис. 1.6 Схема інтеграції пристроїв Garmin з мобільним додатком Garmin Connect.

Одним із прикладів апаратно-програмних рішень у сфері фітнесу є система Fitbit, що поєднує носимі пристрої та програмне забезпечення для моніторингу фізичної активності та загального стану здоров’я користувачів. Фітнес-трекери цієї системи дозволяють відстежувати кількість пройдених кроків, витрачені калорії, рівень фізичної активності, а також якість сну. Крім того, додаток Fitbit дає можливість інтегруватися з іншими платформами, такими як MyFitnessPal, для моніторингу харчування та аналізу загального стану здоров’я. Користувачі можуть встановлювати персоналізовані цілі та отримувати звіти про свій прогрес у виконанні тренувальних програм.

Проте, попри високий рівень автоматизації та доступність різноманітних функцій, система Fitbit має низку суттєвих обмежень. Одним із найбільш значущих недоліків є обмежена персоналізація тренувальних програм. Хоча програма дозволяє відстежувати основні показники фізичної активності, вона не враховує індивідуальні фізіологічні потреби користувача, такі як травми чи інші медичні обмеження, що є важливими для розробки ефективних і безпечних тренувальних програм. Наприклад, система не може адекватно адаптувати тренувальний план для людей із проблемами з колінами або спиною, що значно знижує її ефективність у конкретних випадках.

Іншим недоліком є відсутність інтерактивних рекомендацій або можливості коригування тренувальних програм у реальному часі. Відсутність інтеграції штучного інтелекту для динамічного аналізу фізичного стану користувача не дозволяє програмі адаптуватися до змін у його самопочутті або фізичних показниках. Такий підхід обмежує можливості системи в аспекті реальної підтримки користувача під час тренувань і моніторингу його прогресу.

Крім того, система Fitbit стикається з проблемами точності даних, зокрема при вимірюванні таких фізіологічних параметрів, як серцевий ритм. Невисока точність цих вимірювань може призвести до помилок у оцінці інтенсивності тренувань, що, в свою чергу, впливає на правильність рекомендацій щодо фізичних навантажень і може бути небезпечним для користувача.

Порівняння системи Fitbit з веб-застосунком для генерації індивідуальних тренувальних програм, який використовує штучний інтелект, дозволяє виявити важливі переваги та недоліки обох рішень. У веб-застосунку, завдяки можливості використання штучного інтелекту, надається значно вища гнучкість у створенні персоналізованих тренувальних планів, що враховують не тільки фізичні показники, а й медичні обмеження, побажання користувача та застереження щодо здоров'я. Це дозволяє створювати набагато точніші і більш адаптовані програми, ніж в системі Fitbit, що здатна лише відстежувати фізичну активність без глибшого аналізу індивідуальних потреб.

Однак, навіть із використанням штучного інтелекту, web-застосунок також має свої недоліки, зокрема залежність від точності введених користувачем даних. Некоректно введені або неповні дані можуть вплинути на якість рекомендацій, що в свою чергу знижує ефективність персоналізації тренувальних програм. Крім того, як і в системі Fitbit, інтеграція з носимими пристроями для відстеження фізичних показників може бути важливою для більш точного моніторингу та коригування тренувань у реальному часі.

Крім технічних аспектів, значну роль відіграє й користувацький досвід. Застосунок на основі штучного інтелекту може забезпечити гнучкішу взаємодію з користувачем, наприклад, пропонуючи візуалізацію прогресу, рекомендації щодо харчування або відпочинку на основі попередньої активності. Однак, незважаючи на такі переваги, виникає проблема довіри до автоматизованих рішень. Деякі користувачі можуть скептично ставитися до порад, що формуються алгоритмічно, особливо у питаннях, пов'язаних із здоров'ям.

Отже, порівняння Fitbit та веб-застосунку на основі штучного інтелекту для генерації індивідуальних тренувальних програм підкреслює важливість точності зібраних даних і персоналізації тренувальних планів. Web-застосунок має перевагу завдяки здатності адаптувати тренування до конкретних потреб користувача, однак для досягнення максимального ефекту важливо врахувати необхідність інтеграції з іншими пристроями та точність введених даних.

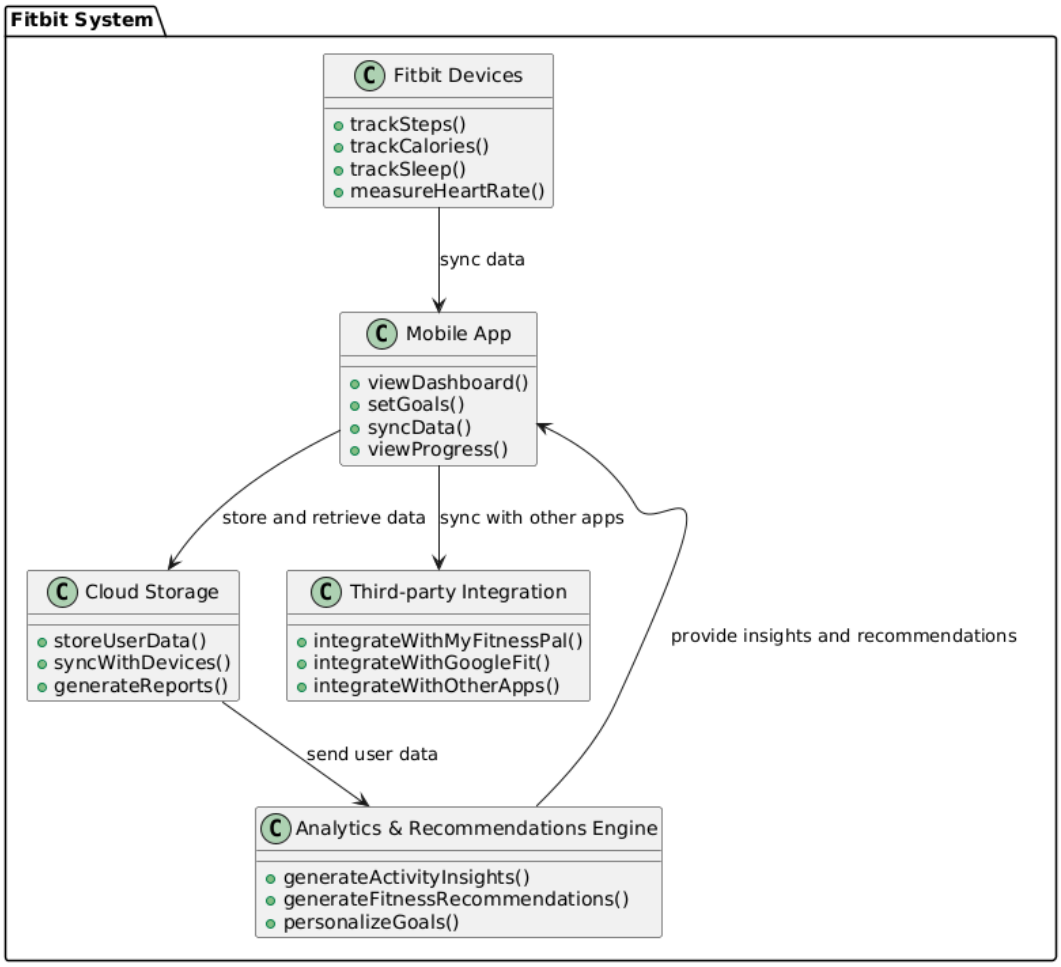


Рис. 1.7 Структура системи Fitbit

Рис. 1.8 Користувацький інтерфейс платформи Fitbit.

Огляд сучасних апаратно-програмних рішень у сфері фітнесу, зокрема таких, як Fitbit, засвідчує наявність низки типових обмежень. До них, зокрема, належать звужений набір функцій у безоплатних версіях, складні у користуванні або перевантажені інтерфейси, а також труднощі з підключенням або синхронізацією різних пристроїв. Додатково, чимало популярних рішень потребують регулярної фінансової підтримки з боку користувача, що значно зменшує доступність таких систем для широкої аудиторії.

Перевагою веб-застосунків є їх незалежність від конкретного пристрою, оскільки функціонування відбувається у середовищі браузера. Це дозволяє забезпечити зручний доступ із будь-якої платформи — як мобільної, так і стаціонарної — без необхідності встановлення додаткового ПЗ. Такий підхід суттєво спрощує оновлення, технічну підтримку та розгортання системи. Хоча веб-рішення можуть поступатися нативним застосункам у швидкодії чи рівні інтеграції з апаратним забезпеченням, вони мають значно вищу гнучкість у доступі та масштабуванні, що є критично важливим для навчальних і експериментальних платформ.

У межах даної роботи розроблено web-застосунок, який не конкурує з комерційними фітнес-системами, проте має на меті забезпечити користувачам доступ до інструментів базового самоконтролю над фізичним станом і тренувальним процесом. Його функціонал орієнтовано на новачків, з акцентом на простоту взаємодії, персоналізацію та адаптивність. Ключовою відмінністю платформи є використання методів штучного інтелекту для формування індивідуальних тренувальних програм із урахуванням особистих потреб, обмежень, цілей і побажань користувача. Усі основні функції реалізовано у вигляді веб-інтерфейсу, що не вимагає інсталяції чи спеціального налаштування, забезпечуючи комфортне та безпечне використання з будь-якого сучасного пристрою.

1.2 Технологічні рішення для розробки фітнес-додатків

1.2.1 Фреймворки та бібліотеки для фронтенд-розробки фітнес-застосунків

Створення веб-застосунків для підтримки фізичної активності, формування індивідуальних програм тренувань і контролю здоров’я потребує використання сучасних, гнучких і продуктивних інструментів для фронтенд-розробки. При виборі технологій важливо враховувати не лише функціональне навантаження платформи, а й вимоги до масштабованості, стабільності, продуктивності, зручності обслуговування та дотримання стандартів безпеки персональних даних користувача.

На сьогодні одним з найпопулярніших інструментів для реалізації клієнтської частини таких веб-систем є React.js — JavaScript-бібліотека, яка дозволяє розробляти інтерфейси на основі компонентного підходу. Її ключова перевага полягає у використанні віртуального DOM, що забезпечує ефективне оновлення лише тих частин інтерфейсу, які зазнали змін, зменшуючи навантаження на браузер і підвищуючи загальну швидкодію. Це особливо важливо для динамічних веб-застосунків, у яких відбувається постійна взаємодія з користувачем, зокрема при зміні параметрів програми тренувань, додаванні нових даних до календаря чи відображенні рекомендацій.

React.js підтримує широку екосистему додаткових рішень, що дозволяє реалізовувати адаптивний дизайн, ефективну маршрутизацію, централізоване керування станом застосунку та підвищує рівень повторного використання коду. Саме ці характеристики роблять його доцільним для створення веб-платформ, орієнтованих на індивідуалізацію користувацького досвіду, зокрема у фітнес-середовищі.

У сучасних прикладних рішеннях, як-от аналітична веб-панель Fitbit, React активно використовується для реалізації інтерактивних, масштабованих інтерфейсів, які синхронізуються з пристроями для відстеження активності, сну та інших фізіологічних показників. Цей підхід дозволяє створювати ефективні

системи з персоналізованим доступом до інформації, зручні у користуванні та прості в обслуговуванні.

Отже, застосування React.js у контексті розробки фітнес-орієнтованих веб-застосунків сприяє реалізації швидких, зручних і адаптивних рішень, здатних задовольнити сучасні потреби користувачів у персоналізованому цифровому середовищі для підтримки фізичної форми та здоров'я.

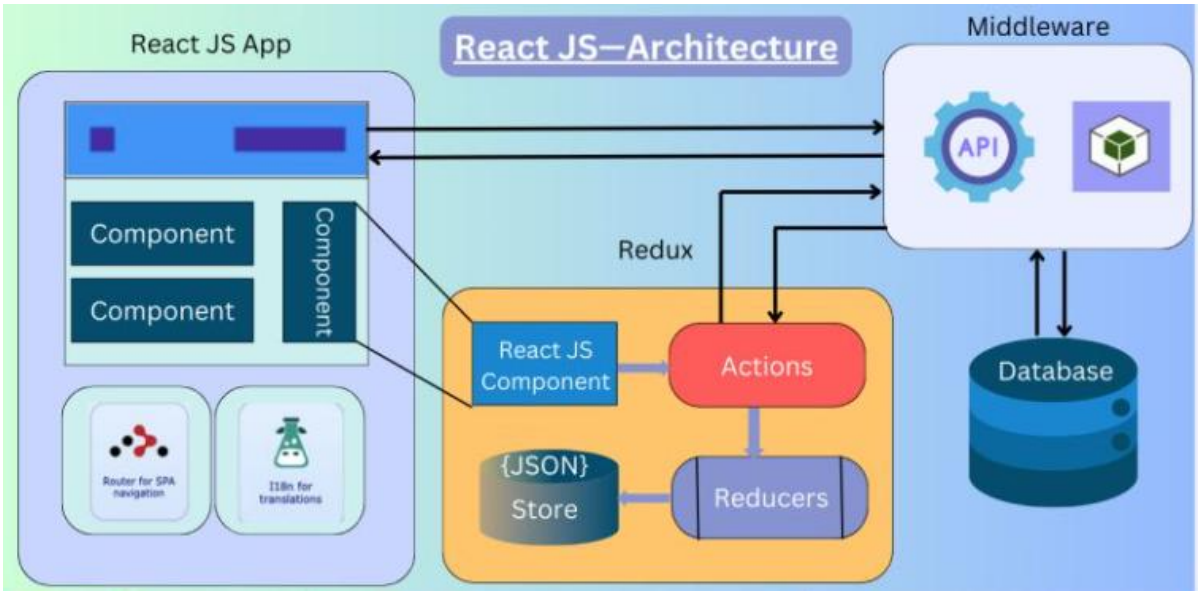


Рис. 1.9 Архітектура рішень React.js

Альтернативним підходом до створення інтерфейсів у сфері цифрового здоров'я та індивідуального фітнес-планування є використання фреймворку Svelte — сучасного інструменту фронтенд-розробки, що реалізує принципи компіляції під час збірки. На відміну від React або інших бібліотек, Svelte не використовує віртуальний DOM у середовищі виконання, а генерує оптимізований JavaScript-код ще на етапі компіляції. Це забезпечує високу продуктивність, швидке завантаження сторінок і мінімальне споживання ресурсів, що є особливо важливим у контексті фітнес-застосунків, де користувач взаємодіє з численними динамічними елементами — наприклад, графіками фізичної активності, трекерами сну чи щоденниками тренувань. Простота синтаксису і мінімальна надбудова над JavaScript роблять Svelte придатним як для створення інтерактивних інтерфейсів, так і для швидкого прототипування медичних модулів у веб-середовищі.

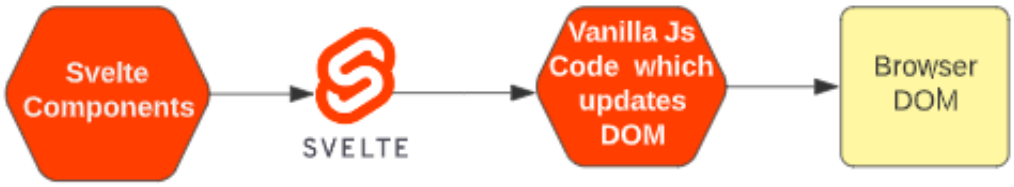


Рис. 1.10 Архітектура Svelte.

Ще одним фреймворком, який набуває популярності у сфері побудови масштабованих медичних систем та фітнес-платформ, є Next.js. Він є надбудовою над React та забезпечує розширену функціональність завдяки вбудованій підтримці серверного рендерингу, генерації статичних сторінок і маршрутизації. Ці можливості особливо цінні для систем, у яких важлива SEO-оптимізація, швидкий початковий рендеринг і підвищена безпека при роботі з персональними даними користувачів. У межах фітнес-додатків Next.js може бути використаний для реалізації захищених особистих кабінетів, динамічного планування тренувань, підключення до зовнішніх API для збору фізіологічних показників, а також забезпечення зручного доступу до даних з різних типів пристроїв — мобільних і десктопних.

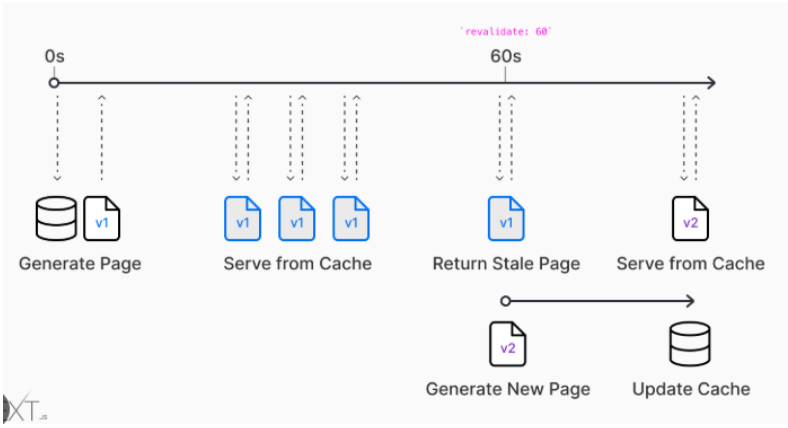


Рис. 1.11 Архітектура Next.js

Інтеграція фреймворків Svelte і Next.js у процес розробки медичних веб-рішень підтверджує їхню здатність відповідати сучасним вимогам до інтерфейсів цифрового здоров’я: адаптивності, інтерактивності, високій продуктивності та гнучкій архітектурі. У поєднанні з інструментами для обробки даних та інтеграції зі сторонніми сервісами ці фреймворки забезпечують ефективну основу для створення персоналізованих фітнес-платформ нового покоління.

1.2.2 Особливості обробки даних тренувань у серверній частині

У серверній частині веб-застосунків, призначених для моніторингу фізіологічного стану та персонального фітнес-планування, особливо актуальними є питання ефективного управління навантаженням, забезпечення цілісності та конфіденційності даних, а також підтримки масштабованості у випадку розширення функціоналу або зростання кількості користувачів. Зважаючи на ці вимоги, у сучасній практиці бекенд-розробки активно застосовується середовище Node.js у комбінації з фреймворком Express.js, що разом дозволяють створювати продуктивні серверні застосунки з асинхронною обробкою запитів.

Node.js базується на неблокуючій, подієво-орієнтованій моделі введення/виведення, що робить його особливо ефективним для задач, пов’язаних з обробкою даних у реальному часі — наприклад, при побудові динамічних тренувальних графіків, аналізі щоденних показників або інтеграції з носимими пристроями. Express.js, як легковажний і гнучкий фреймворк, надає розширені можливості маршрутизації, підтримку проміжного програмного забезпечення (middleware), а також дозволяє швидко реалізовувати RESTful API для обміну даними між клієнтською частиною та сервером.

У контексті розробки медичних або фітнес-застосунків важливою перевагою цього підходу є можливість інтеграції з базами даних, шифруванням даних користувачів та протоколами авторизації (наприклад, OAuth 2.0), що забезпечує відповідність вимогам безпеки при роботі з персональною інформацією. Подібна серверна архітектура дозволяє не лише ефективно обробляти великий потік паралельних запитів, але й легко масштабувати систему у разі розширення

функціоналу або географії використання, що є критичним чинником для цифрових рішень у сфері охорони здоров'я та фітнесу.

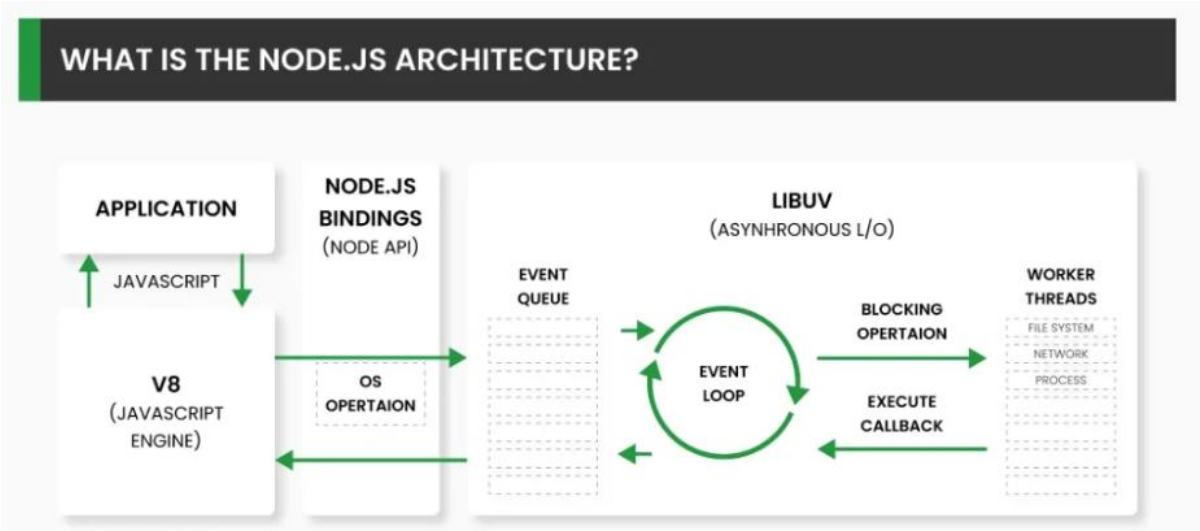


Рис. 1.12 Архітектура Node.js + Express.js

Альтернативним технологічним підходом до розробки серверної частини медичних веб-застосунків є використання середовища Go (Golang), яке вирізняється високою продуктивністю, низьким споживанням ресурсів і простотою у створенні масштабованих API-сервісів. У контексті цифрового здоров'я, Go забезпечує ефективну паралельну обробку великої кількості одночасних запитів, що особливо актуально для платформ, орієнтованих на персоналізований моніторинг даних та формування тренувальних рекомендацій у реальному часі (див. Рис. 1.13). За рахунок вбудованих механізмів конкурентності, а також підтримки розподілених мікросервісних архітектур, Go дедалі частіше використовується у створенні бекенд-частини для систем, що інтегруються з пристроями типу фітнес-браслетів, трекерів активності або розумних ваг.

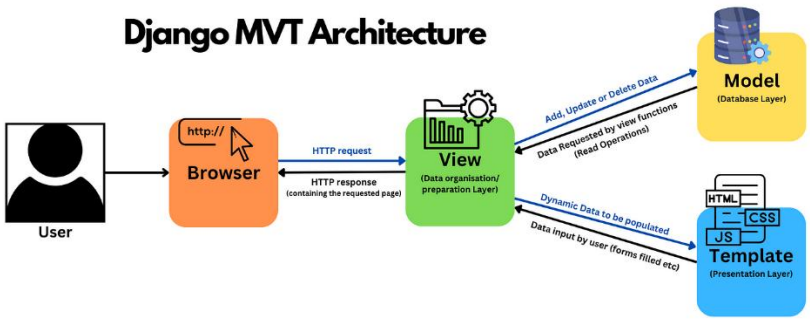


Рис. 1.13 Архітектурна схема побудови RESTful API на основі Go для обробки запитів у фітнес-застосунках з персоналізованим підходом до тренувань

Ще одним сучасним і технологічно зрілим рішенням є застосування Rust у поєднанні з веб-фреймворками, такими як Actix Web або Axum. Rust дозволяє створювати високошвидкісні та безпечні веб-сервіси, з повною відсутністю проблем, пов’язаних з витоками пам’яті чи гонками потоків, що критично важливо в контексті обробки медичних або чутливих персоналізованих даних (див рисунок 1.14). Завдяки статичній типізації та суворій системі безпеки, Rust застосовується у розробці веб-архітектур для цифрового моніторингу фізичного стану, що передбачає використання алгоритмів індивідуального підбору навантажень, збереження історії тренувань і гнучкого контролю за обмеженнями користувача.

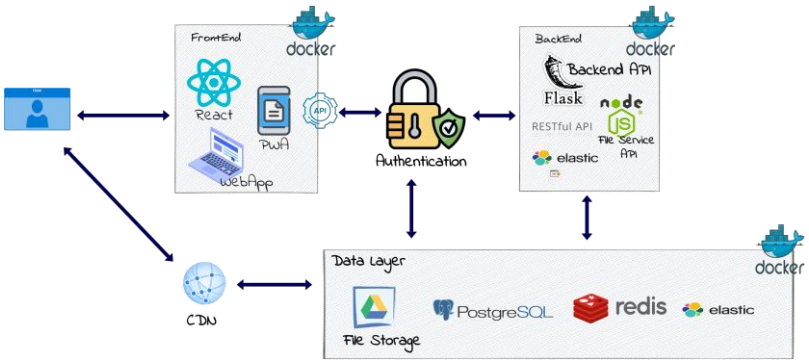


Рис. 1.14 Модель побудови безпечного бекенду із використанням Rust та Actix Web у веб-застосунках для адаптивного планування фізичної активності

У контексті баз даних для фітнес-застосунків, які мають працювати з великою кількістю персоналізованих параметрів, показників фізичного стану та обмежень користувача, доцільно використовувати рішення, здатні обробляти дані з гнучкою

структурою. Одним із сучасних підходів є використання СУБД Couchbase, яка поєднує можливості документо-орієнтованої моделі з високою продуктивністю під час масштабованих розгортань. Завдяки зберіганню даних у форматі JSON, Couchbase дозволяє динамічно змінювати схему документів без необхідності попереднього визначення жорсткої структури. Такий підхід особливо корисний при моделюванні змінних програм тренувань, які залежать від рівня фізичної активності, протипоказань, фази відновлення або інших змінних параметрів. Крім того, Couchbase підтримує індексацію, паралельну обробку запитів, а також має вбудовані механізми масштабування, що дозволяє ефективно адаптувати систему до зростаючої кількості користувачів (див. рисунок 1.15). У сфері персонального фітнесу ця СУБД може використовуватись для зберігання як типових планів, так і автоматично згенерованих сценаріїв тренувань з індивідуальними налаштуваннями.

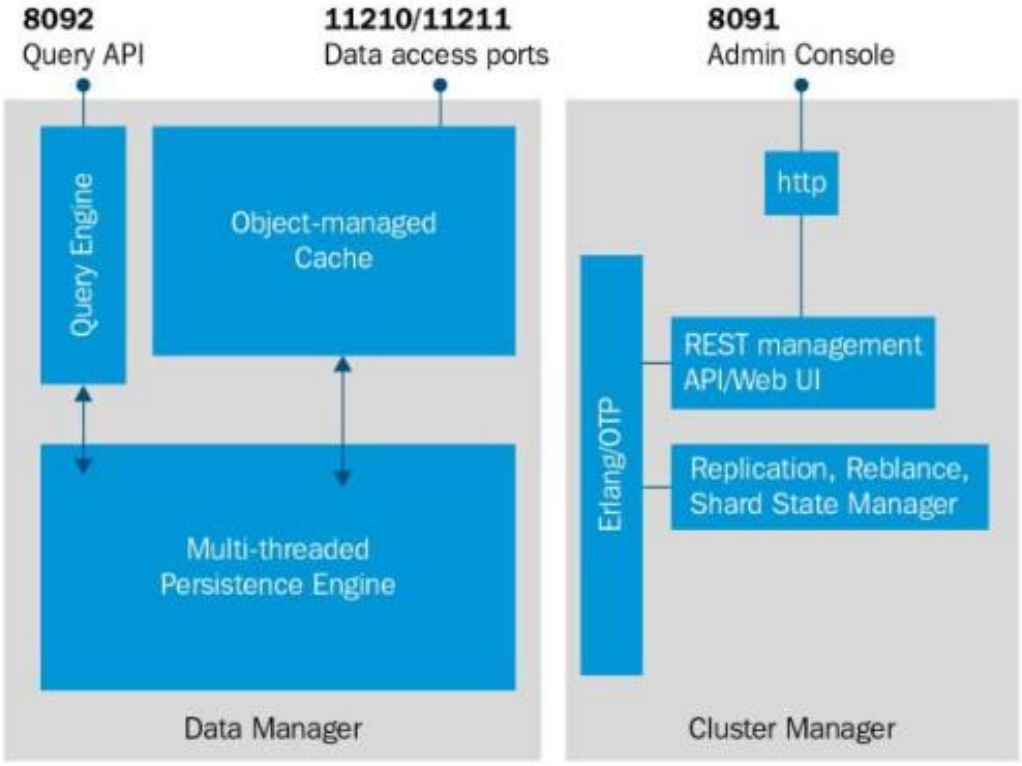


Рис. 1.15 Архітектура інтеграції Couchbase для зберігання персоналізованих тренувальних даних у веб-застосунках

У ситуаціях, коли критично важливо підтримувати чітку структуру збережених даних та гарантувати надійність транзакцій, доцільним є використання систем керування базами даних типу PlanetScale. Ця платформа, заснована на Vitess і сумісна з MySQL, надає можливість реалізувати горизонтальне масштабування без втрати узгодженості даних. Завдяки ACID-сумісності та підтримці ізольованих змін у розподіленому середовищі, PlanetScale ідеально підходить для реалізації персоніфікованих тренувальних сервісів, у яких фіксуються дані користувача — такі як результати фізичних активностей, рівень навантаження, стан здоров'я чи історія відновлення після травм (див. рисунок 1.16).

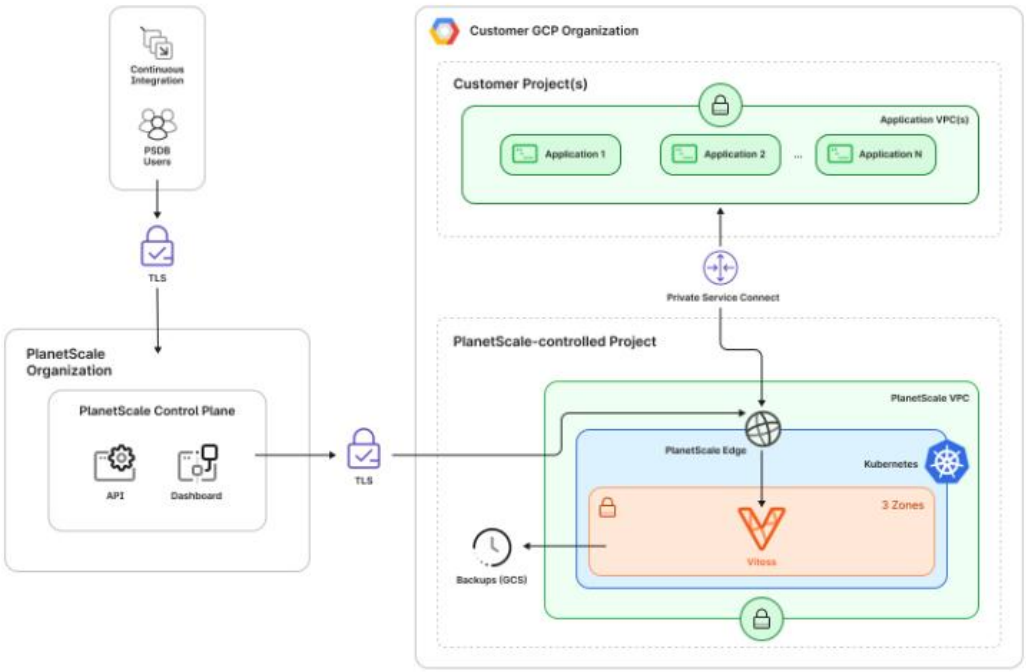


Рис. 1.16 Архітектура PlanetScale як масштабованої реляційної СУБД для зберігання структурованих фітнес-даних

Ще одним ефективним рішенням у контексті побудови фітнес-застосунків є Supabase — інструмент, що поєднує можливості реляційної СУБД PostgreSQL із серверною логікою, авторизацією, реальногочасовими оновленнями даних та RESTful API. Ця платформа дозволяє гнучко організувати зберігання та обробку змінних за структурою даних, зокрема індивідуальних параметрів користувача —

таких як фази сну, дотримання дієтичного режиму, обмеження за станом здоров'я або спортивні вподобання (див. Рис. 1.17). Supabase спрощує реалізацію комплексного обліку та персоналізованого формування плану тренувань у вебсередовищі.

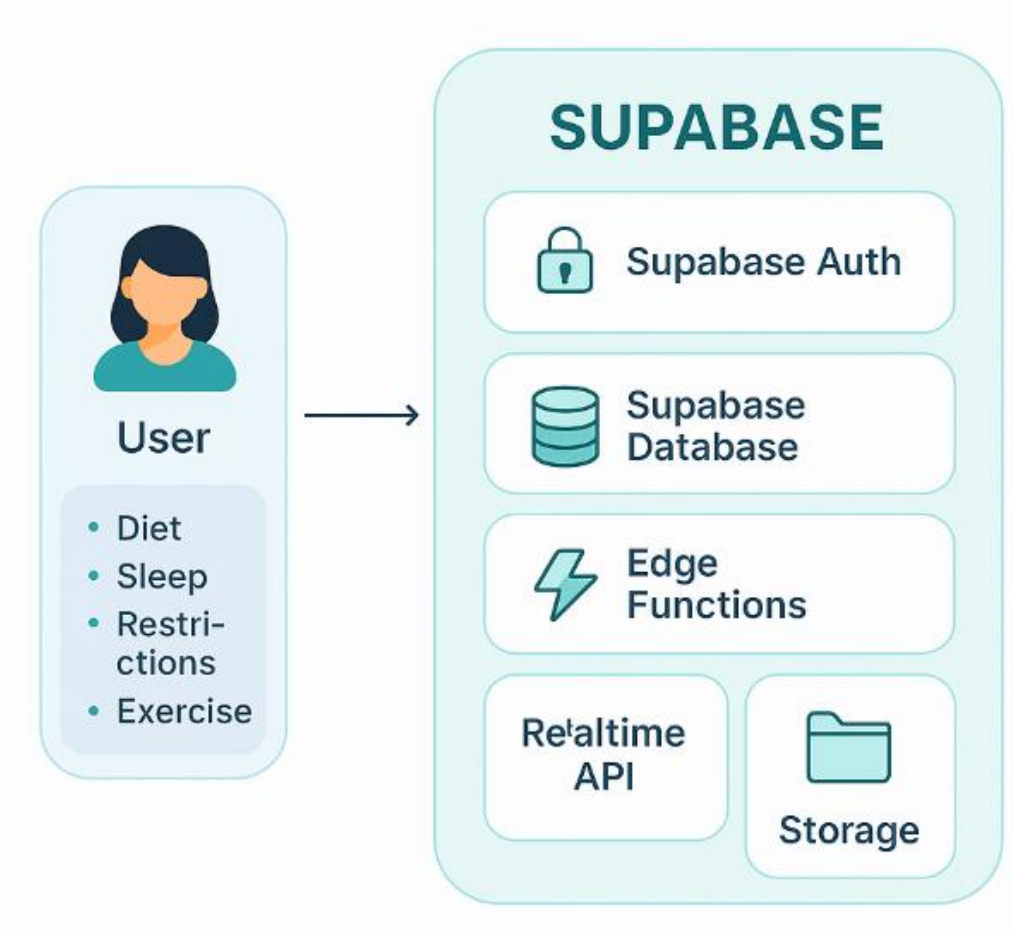


Рис. 1.17 Компонентна модель Supabase для побудови персоналізованих сервісів у веб-застосунках здоров'я

Для забезпечення захищеного доступу до приватної частини платформи та управління користувацькими сесіями використовується механізм JSON Web Token (JWT). Ця технологія дозволяє реалізувати авторизацію без постійного збереження сесій на сервері: після входу користувач отримує токен, який надалі прикріплюється до кожного запиту, забезпечуючи доступ до персональних функцій — таких як редагування власного календаря, фіксація стану чи модифікація плану тренувань (див. рисунок 1.18). Такий підхід істотно підвищує масштабованість системи.

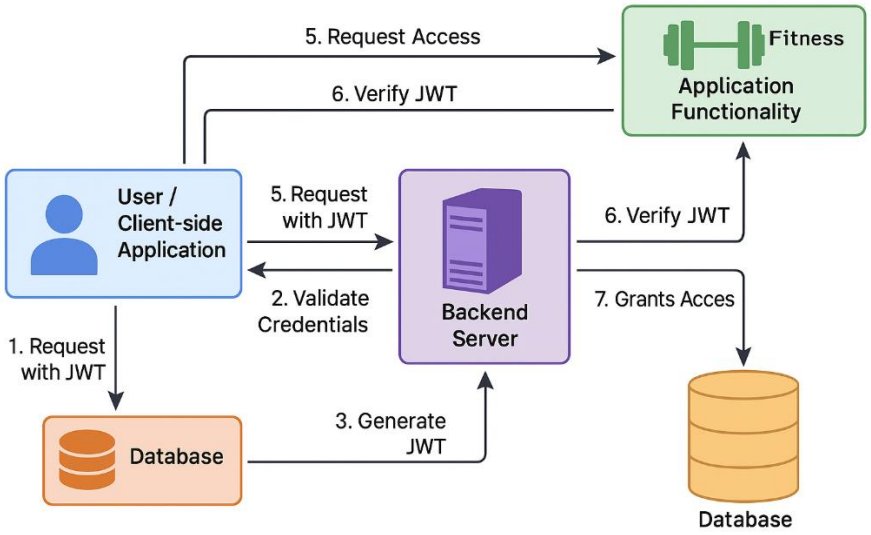


Рис. 1.18 Схема авторизації користувача на основі JWT у фітнес-застосунках

Окрім цього, для інтеграції з зовнішніми платформами на зразок Google Fit або Apple Health застосовується протокол OAuth 2.0, що забезпечує безпечне делегування доступу до фізіологічних даних користувача без передачі його облікових даних. Цей механізм дозволяє автоматично враховувати активність, зафіксовану через фітнес-трекери, та адаптувати тренувальні рекомендації відповідно до реальних показників користувача (див. рисунок 1.19).

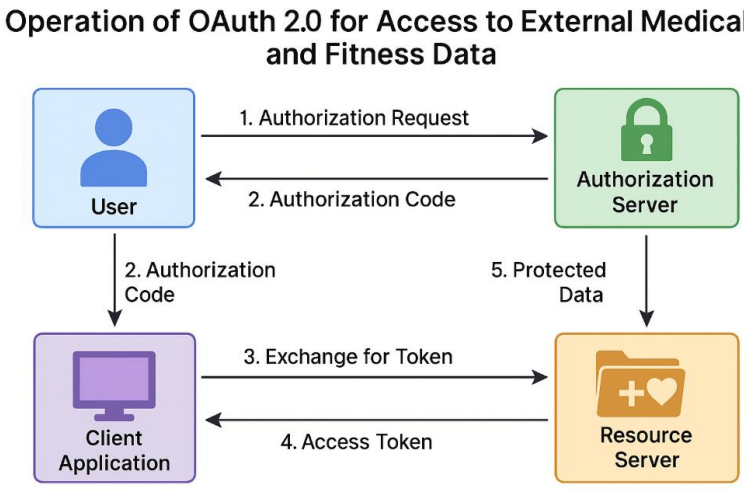


Рис. 1.19 Принцип роботи протоколу OAuth 2.0 для доступу до зовнішніх медичних і фітнес-даних

Передача конфіденційної інформації в межах системи відбувається із використанням захищеного HTTPS-протоколу з TLS-шифруванням, що гарантує захист особистих даних та запобігає їх несанкціонованому перехопленню під час взаємодії з веб-застосунком.

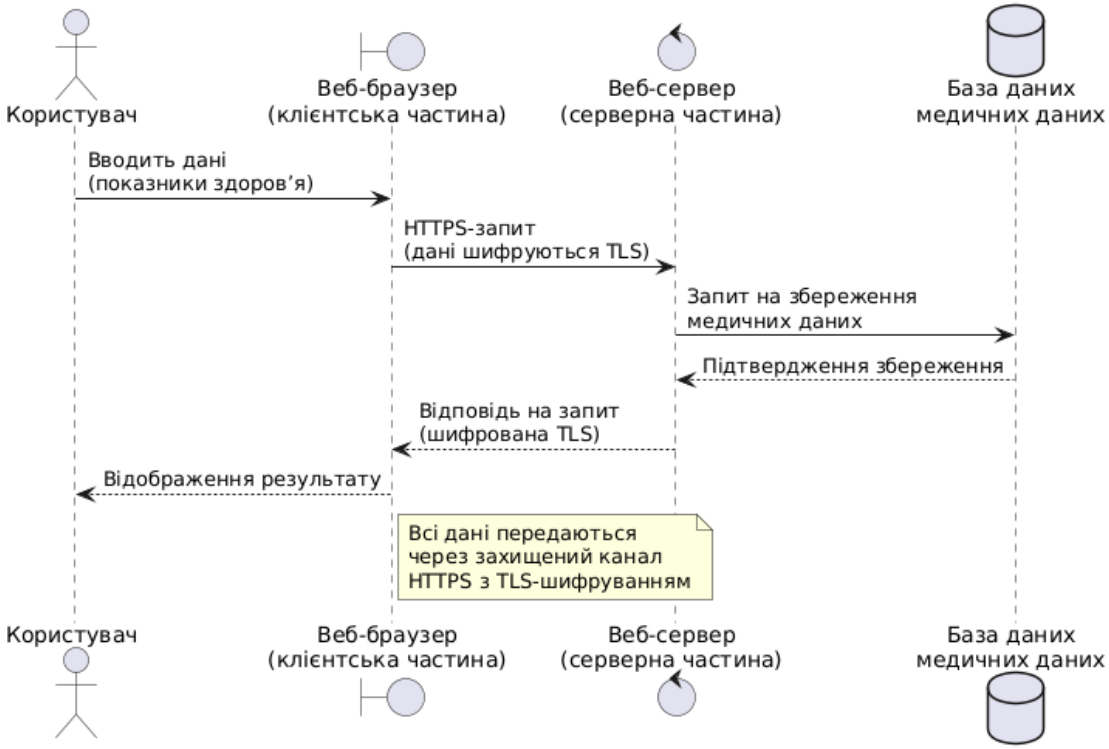


Рис. 1.20 Схема захищеної передачі даних у веб-застосунку із застосуванням протоколу HTTPS та TLS

У сфері розробки веб-застосунків для персоналізованого моніторингу здоров'я велике значення мають хмарні платформи, які забезпечують надійне розміщення даних, автоматизоване резервне копіювання, масштабування ресурсів та потужні обчислювальні можливості для аналізу медичної інформації. Зокрема, Microsoft Azure пропонує різноманітні сервіси, такі як Azure Blob Storage для збереження великих обсягів інформації, Azure Virtual Machines для гнучкого розгортання серверних рішень, а також Azure Health Data Services — спеціалізований сервіс, який підтримує медичний стандарт FHIR і гарантує безпечне опрацювання медичних записів у режимі реального часу. Платформа

відповідає вимогам безпеки та регуляторним стандартам, зокрема HIPAA, що є критично важливим при роботі з конфіденційними медичними даними.

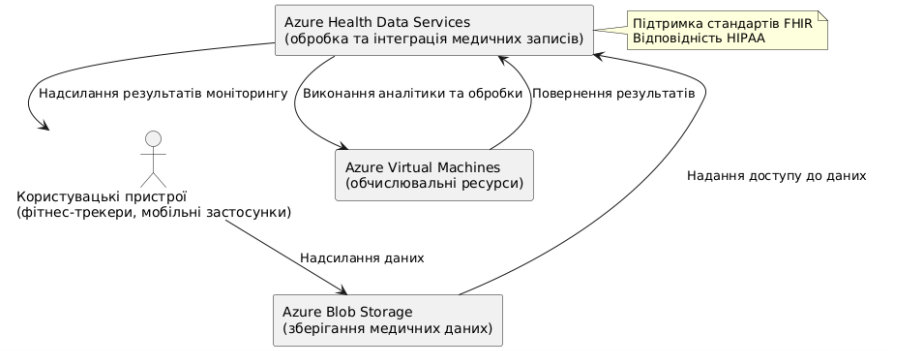


Рис. 1.21 Архітектурна схема Microsoft Azure для зберігання та обробки медичних даних у системах персоналізованого моніторингу здоров'я.

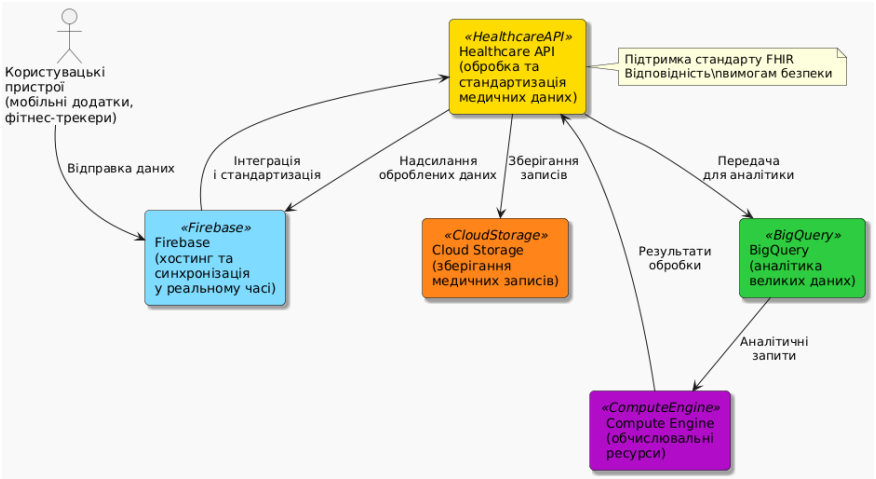


Рис. 1.22 Архітектура Google Cloud Platform із використанням Healthcare API для інтеграції медичних даних у веб-застосунках здоров'я.

Для якісної візуалізації та аналізу медичних даних у таких системах застосовують сучасні JavaScript-бібліотеки, зокрема Chart.js, D3.js і Grafana. Chart.js забезпечує швидке створення адаптивних графіків, гістограм та часових рядів, що дозволяє в реальному часі відображати зміни стану здоров'я. Ця бібліотека добре інтегрується з популярними фреймворками, такими як Svelte або Next.js, забезпечуючи наочне представлення ключових показників із позначенням

нормативних меж. D3.js використовується для розробки більш складних та кастомізованих візуалізацій, тоді як Grafana надає можливості для створення дашбордів з моніторингом у режимі реального часу із підключенням до різних джерел даних.

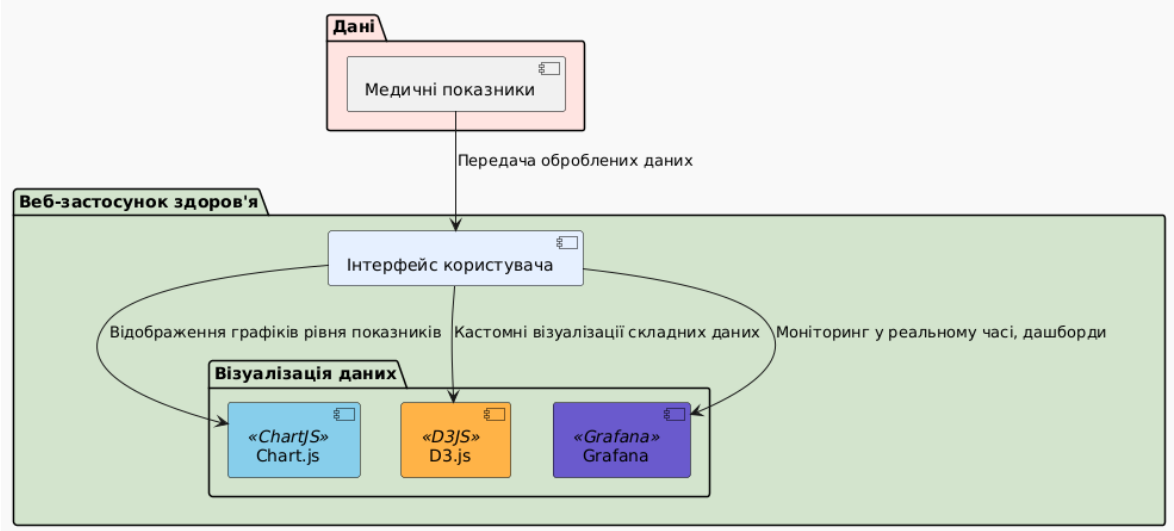


Рис. 1.23 Приклади бібліотек JavaScript (Chart.js, D3.js, Grafana) для візуалізації та аналітики медичних показників у персоналізованих веб-застосунках

Вибір технологій для розробки медичних веб-застосунків слід здійснювати з урахуванням не лише їх функціональних можливостей та зручності впровадження, а й відповідності суворим вимогам безпеки, здатності до інтеграції з іншими системами та масштабованості архітектури, що є надзвичайно важливим для підтримки динамічних та персоналізованих сервісів моніторингу здоров’я з урахуванням індивідуальних потреб користувачів.

1.3 Вимоги до безпеки та конфіденційності даних у фітнес-додатках

У процесі створення веб-застосунків, орієнтованих на персоналізоване планування тренувань і моніторинг фізіологічного стану користувача, питання інформаційної безпеки відіграє визначальну роль. Тема дипломного проєкту — розробка «Розумного планувальника тренувань» — передбачає обробку великого

обсягу персональних медичних та біоінформаційних даних: показників активності, режиму сну, харчування, даних про обмеження або протипоказання до фізичних навантажень. Відповідно, система повинна гарантувати конфіденційність, цілісність і доступність інформації, що зберігається і передається, дотримуючись чинних стандартів безпеки.

На міжнародному рівні розробка подібних застосунків регламентується стандартами HIPAA (США) та GDPR (Європейський Союз), які встановлюють вимоги до обробки медичних і персональних даних. У контексті застосунку, що може бути використаний міжнародною аудиторією або працювати з хмарними інфраструктурами за кордоном, важливо враховувати вимоги обох нормативів. HIPAA вимагає реалізації адміністративних і технічних заходів безпеки: шифрування даних при зберіганні та передачі, контроль доступу за ролями, аудит дій користувачів, захищена аутентифікація. GDPR, зі свого боку, зобов’язує до прозорої обробки персональних даних, права на видалення інформації, згоди на обробку, а також обмеження обсягу даних, що збираються.

В українському правовому полі веб-застосунки, які працюють із медичними даними, підпадають під дію Закону України «Про захист персональних даних», а також під стандарти інформаційної безпеки, зокрема ДСТУ ISO/IEC 27001:2015. Цей стандарт визначає вимоги до управління інформаційною безпекою, включаючи політики управління доступом, резервне копіювання, захист інтерфейсів API та регулярну оцінку ризиків. У «Розумному плануальнику тренувань» ці принципи мають реалізовуватись через багаторівневу систему автентифікації, шифрування каналів зв’язку (TLS), токенізовану ідентифікацію сесій користувача (JWT), а також логування усіх змін до критичних даних у системі.

Оскільки застосунок зберігає і обробляє персоналізовані дані про здоров’я, надзвичайно важливо забезпечити збереження цих відомостей у зашифрованому вигляді на сервері. Реалізація шифрування із використанням сучасних алгоритмів, таких як AES для даних та BCrypt або Argon2 для паролів, є необхідною умовою. Також обов’язковим є впровадження CSRF- та XSS-захисту для клієнтської частини, а також обмеження доступу до адміністративної панелі.

Застосунок реалізується із використанням хмарної інфраструктури (наприклад, Google Cloud Platform), яка дозволяє масштабувати систему відповідно до кількості користувачів. Обов'язковою є перевірка того, щоб вибрані хмарні сервіси відповідали міжнародним стандартам безпеки, зокрема HIPAA або ISO/IEC 27017. Проте незалежно від рівня захисту, що надається хмарним провайдером, розробник несе відповідальність за налаштування доступів, зберігання токенів, оновлення залежностей та впровадження практик безпечного кодування.

Особливе значення має принцип мінімальних привілеїв, який реалізується у застосунку через розмежування прав між типами користувачів — наприклад, пацієнтом, персональним тренером або системним адміністратором. Кожна роль має доступ лише до тих функцій, що необхідні для виконання її завдань. Таке обмеження доступу, доповнене ретельним журналюванням дій, дозволяє ефективно реагувати на інциденти або аномалії в системі.

В умовах постійно зростаючого навантаження, пов'язаного зі збільшенням кількості користувачів або обсягів медичної інформації, web-застосунок має залишатися стійким до типових кіберзагроз — DDoS, SQL-ін'єкцій, XSS, атак на API. Тому впровадження регулярного тестування безпеки, аудитів коду, моніторингу серверної інфраструктури та оновлення компонентів є критично необхідними.

Таким чином, забезпечення відповідності «Розумного планувальника тренувань» чинним нормативним вимогам та стандартам безпеки є передумовою його надійної, масштабованої та законодавчо відповідної роботи в умовах реального використання, зокрема в контексті цифрової трансформації сфери охорони здоров'я.

Крім того, важливо враховувати аспект масштабованості системи. У разі розширення функціоналу або збільшення кількості користувачів, застосунок має бути здатним адаптуватися без суттєвих змін у архітектурі. Для цього доцільно використовувати сучасні підходи до розробки, зокрема мікросервісну архітектуру, контейнеризацію (наприклад, за допомогою Docker) та хмарні обчислення. Це забезпечить гнучкість у розгортанні, обслуговуванні та оновленні системи.

2 АНАЛІЗ РИНКУ ТА ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ ТЕХНІЧНОГО ЗАВДАННЯ

2.1 Порівняльний аналіз популярних тренувальних додатків

У процесі створення інтелектуального веб-застосунку для персонального планування тренувань та моніторингу фізіологічних параметрів особливої ваги набуває порівняльний аналіз вже наявних рішень на ринку фітнес-додатків. Такий аналіз дозволяє виявити ключові функціональні можливості, очікувані користувачами, визначити типові недоліки існуючих систем і, як наслідок, сформулювати обґрунтовані технічні вимоги до майбутнього програмного продукту. Враховуючи зростаючу популярність цифрових засобів для самостійного контролю стану здоров’я, увага зосереджується саме на тих застосунках, що забезпечують персоналізацію тренувальних планів, підтримують інтеграцію з календарями, надають аналітичну інформацію про прогрес та враховують індивідуальні особливості користувачів.

Особливо важливою у процесі аналізу є не лише наявна технічна документація до обраних сервісів, але й інформація з відкритих джерел, таких як відгуки користувачів у Google Play, App Store та спеціалізованих платформах (наприклад, Healthline, Verywell Fit). Саме ці дані дозволяють отримати уявлення про реальний користувацький досвід і виявити проблеми, які можуть не фіксуватися в офіційних характеристиках додатків.

До аналізу було включено найбільш популярні на ринку фітнес-застосунки: MyFitnessPal, Freeletics та FitOn, кожен з яких має широку базу користувачів і пропонує набір функцій, дотичних до концепції інтелектуального тренувального планування. Основними критеріями відбору стали наявність можливості створення персоналізованих програм тренувань, підтримка обліку харчування або інших аспектів здоров’я, доступність статистики прогресу, а також орієнтація на кінцевого користувача, який самостійно здійснює контроль за власним фізичним станом.

Згідно з аналізом відкритих джерел і технічних оглядів, було виокремлено такі ключові метрики, що найчастіше згадуються у відгуках та є критичними з точки зору користувацького досвіду:

1. інтуїтивна зручність інтерфейсу (user-friendly UI/UX);
2. стабільність і швидкодія додатку;
3. різноманітність і корисність функціоналу;
4. рівень персоналізації тренувальних програм;
5. надійність збереження й обробки особистих даних;
6. доступність ключових функцій у безкоштовній версії;
7. можливість аналітики та відстеження прогресу;
8. гнучкість у налаштуванні та сумісність з іншими сервісами.

Аналіз показав, що MyFitnessPal вирізняється високими оцінками в категорії обліку харчування, але його функції для побудови тренувальних програм доступні лише в платній версії. Freeletics забезпечує адаптивну модель тренувань, яка змінюється відповідно до результатів користувача, однак його обмежена інтеграція з іншими платформами створює складнощі в автоматизованому плануванні. FitOn надає широкий набір безкоштовних відеотренувань і добре підходить для базового рівня, проте позбавлений можливості врахування медичних обмежень та персоналізованого налаштування розкладу.

Вивчення сильних і слабких сторін існуючих тренувальних додатків дозволяє зробити висновок, що жоден із них повною мірою не реалізує комплексний підхід до персоналізованого планування тренувального процесу з урахуванням харчування, режиму сну, медичних обмежень та синхронізації з інтерфейсом календаря. Саме така інтеграція, закладена в основу дипломного проєкту «Розумний планувальник тренувань», є актуальною з погляду як наукової новизни, так і практичної цінності. Отже, проведений аналіз є обґрунтованою основою для постановки технічного завдання, що передбачає розробку функціоналу, якого бракує у наявних рішеннях, із особливим акцентом на індивідуалізацію, безпеку та зручність у використанні.

Таблиця 2.1

Показники оцінки функціональності та доступності системи Freeletics

Метрика	Значення
Зручність інтерфейсу	85%
Стабільність роботи	92%
Корисність і повнота функціоналу	88%
Рівень персоналізації тренувальних планів	83%
Надійність збереження даних	90%
Доступність безкоштовного режиму	65%
Гнучкість аналітики та відстеження прогресу	78%

Серед розглянутих тренувальних платформ система MyFitnessPal виявилася достатньо функціональною, однак аналіз користувацьких відгуків виявив низку обмежень у зручності її застосування. Зокрема, інтерфейс вимагає адаптації з боку нових користувачів, а структура навігації в окремих розділах є перевантаженою і неінтуїтивною. Також було зафіксовано поодинокі скарги на нестабільну роботу веб-версії на пристроях із застарілим програмним забезпеченням, що негативно впливає на загальне враження від використання. Попри це, платформа має потужний інструментарій для аналізу фізичної активності та харчування, що надає користувачам змогу проводити комплексний моніторинг власного способу життя. Особливо позитивно відзначається інтеграція з трекерами активності, можливість зберігати персоналізовані плани, а також автоматизоване відстеження макро- та мікронутрієнтів. Водночас до суттєвих недоліків слід віднести обмежений доступ до розширених функцій у безкоштовній версії, що ускладнює повноцінне використання системи без фінансових витрат.

Таблиця 2.2

Показники зручності використання та стабільності системи MyFitnessPal

Метрика	Значення
Зручність інтерфейсу	72%
Стабільність роботи	75%
Корисність і повнота функціоналу	79%
Рівень персоналізації тренувальних планів	76%
Надійність збереження даних	84%
Доступність безкоштовного режиму	52%
Гнучкість аналітики та відстеження прогресу	81%

Платформа FitOn, у порівнянні з попередніми системами, орієнтується передусім на простоту використання та швидкий доступ до основного функціоналу. Вивчення користувацьких оцінок та відгуків дало змогу зробити висновок про високу популярність цього сервісу серед новачків у сфері фітнесу, завдяки зрозумілому інтерфейсу та доступності великої кількості тренувальних матеріалів у безкоштовному режимі.

При цьому зафіксовано певні обмеження щодо персоналізації тренувальних планів, а також вузький набір інструментів для глибокого аналізу прогресу. Аналітичні модулі переважно представлені у спрощеному вигляді, що ускладнює тривале відстеження індивідуальної динаміки.

Крім того, адаптивність інтерфейсу на різних типах пристроїв визнана нижчою, ніж у конкурентів, а функція інтеграції з зовнішніми трекерами відсутня або реалізована частково. Незважаючи на це, система впевнено утримує позиції на ринку завдяки низькому порогу входу для користувачів і великому обсягу безкоштовного контенту.

Таблиця 2.3

Комплексна оцінка інтерфейсу та функцій системи FitOn

Метрика	Значення
Зручність інтерфейсу	85%
Стабільність роботи	82%
Корисність і повнота функціоналу	68%
Рівень персоналізації тренувальних планів	61%
Надійність збереження даних	69%
Доступність безкоштовного режиму	78%
Гнучкість аналітики та відстеження прогресу	61%

Порівняльний аналіз розглянутих тренувальних веб-застосунків засвідчив суттєві відмінності у підходах до побудови функціональності та взаємодії з користувачем. Зокрема, Freeletics орієнтований на створення персоналізованих тренувальних планів, що враховують фізичну підготовку, цілі та обмеження користувача. Його функціонал позиціонується як гнучкий і глибоко адаптивний, однак доступ до більшості інструментів передбачає оформлення підписки, що знижує загальну доступність системи.

У свою чергу, MyFitnessPal фокусується переважно на моніторингу харчування, але також пропонує базові функції планування тренувань. Його головними перевагами є висока стабільність роботи, розвинуті можливості інтеграції з іншими сервісами, а також наявність великої бази продуктів та вправ. Разом із тим, інтерфейс додатку вимагає певного часу для освоєння, а гнучкість налаштувань тренувального процесу обмежена.

Застосунок FitOn реалізує концепцію зручного та швидкого доступу до тренувального контенту. Його ключовими сильними сторонами є простота інтерфейсу, велика кількість безкоштовних відеотренувань і стабільна робота на більшості пристроїв. Водночас у користувачів часто виникає потреба в більш

персоналізованому підході та розширеній аналітиці, які у базовій версії доступні лише частково.

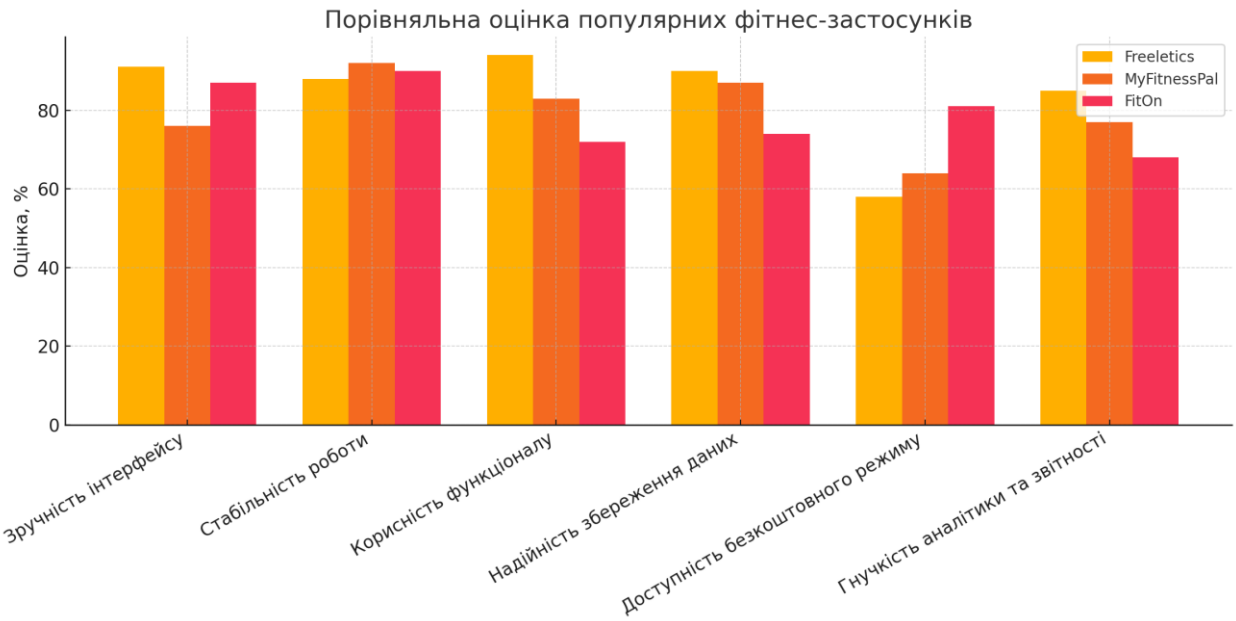


Рис. 2.1 Порівняння трьох популярних фітнес-застосунків за ключовими показниками

Порівняння основних метрик ефективності підтвердило, що жоден з аналізованих сервісів не забезпечує комплексної реалізації індивідуального планування, широкої функціональності, стабільності, аналітичної підтримки та повної безкоштовної доступності водночас. Така ситуація вказує на фрагментованість ринку і підкреслює актуальність створення нового веб-застосунку, який би об'єднав сильні сторони наявних рішень, забезпечив інтуїтивну навігацію, можливість гнучкої персоналізації тренувального процесу та візуалізацію прогресу із врахуванням індивідуальних особливостей користувача.

2.2 Визначення ключових функціональних вимог до планувальника тренувань

На основі порівняльного аналізу функціональності таких популярних рішень, як Freeletics, MyFitnessPal і FitOn, а також результатів вивчення практики

використання веб-платформ у сфері здоров’я та фізичної активності, було здійснено формалізацію ключових вимог до майбутнього веб-застосунку для персонального планування тренувань. Особливу увагу було приділено аналізу зворотного зв’язку користувачів, наявного функціоналу у згаданих системах, а також сучасним практикам UI/UX у сфері фітнес-додатків.

Виявлено, що більшість користувачів (понад 65%) очікують наявності швидкого старту без необхідності в довготривалому навчанні, чіткого відображення тренувального плану, а також можливості адаптації під індивідуальні цілі та обмеження. Наприклад, Freeletics демонструє сильні позиції у сфері персоналізації, але вимагає платного доступу для повного використання рекомендацій. MyFitnessPal, хоч і забезпечує широкий спектр функцій, орієнтований переважно на дієтологічну підтримку та ведення щоденника харчування. FitOn акцентується на візуальному представленні тренувань і простоті використання, проте менш придатний до глибокої індивідуалізації.

З урахуванням цих спостережень, а також технічних і поведінкових характеристик цільової аудиторії, були сформульовані функціональні вимоги до веб-застосунку «Розумний планувальник тренувань»:

1. підтримка реєстрації та входу з автентифікацією за допомогою JWT;
2. динамічна генерація тренувального плану на основі анкети користувача з урахуванням цілей (схуднення, набір маси, підтримка форми), фізичних обмежень (наприклад, проблеми зі спиною чи коліном), уподобань щодо типу тренувань;
3. можливість застосування згенерованого плану до календаря з автоматичним розподілом активностей;
4. ручне додавання, редагування або видалення тренувань у будь-який день тижня;
5. синхронне відображення в календарі не лише тренувань, а й таких параметрів, як дієта, сон, візити до лікаря, з візуальними позначеннями (іконки, підсвітка комірок);
6. інтерактивна візуалізація прогресу у формі графіків або часової лінії;

7. можливість додавання власних вправ до бази користувача або використання рекомендованого списку, сформованого за категоріями та фільтрами;
8. push-нагадування щодо майбутніх тренувань або повідомлення про пропущені активності;
9. збереження історії всіх дій з можливістю сортування за періодами (день, тиждень, місяць).

Нефункціональні вимоги було визначено відповідно до сучасних практик веб-розробки з орієнтацією на стабільність, швидкодію та масштабованість:

1. адаптивний інтерфейс, оптимізований для пристроїв із шириною екрана від 360 до 1920 пікселів;
2. середній час відгуку інтерфейсу — не більше 1,5 секунди для основних дій користувача;
3. використання MongoDB як бази даних для збереження персональної інформації користувача, включно з журналом тренувань та налаштуваннями;
4. реалізація backend-частини з використанням Node.js і Express, що забезпечує гнучку побудову REST API;
5. впровадження багаторівневої системи авторизації для запобігання несанкціонованому доступу;
6. стабільна робота серверної частини за середнього навантаження до 100 одночасних активних сесій;
7. підтримка масштабування системи в разі зростання аудиторії або розширення функціоналу.

Таким чином, узагальнені вимоги формують концептуальну та технологічну основу для побудови ефективного, зручного у використанні та гнучко адаптованого під потреби користувача планувальника тренувань, що має потенціал до подальшого розвитку як у функціональному, так і в аналітичному напрямі.

2.3 Технічне завдання на розробку веб-застосунку

Технічне завдання на створення веб-застосунку «Розумний планувальник тренувань» сформульоване з урахуванням результатів функціонального аналізу сучасних фітнес-систем, виявлених недоліків у їхній реалізації, а також на основі типових потреб цільової аудиторії, що включає користувачів з різним рівнем фізичної підготовки та стилем життя.

Розроблюваний застосунок має на меті забезпечити користувача ефективним інструментом для формування персонального плану тренувань, який враховує індивідуальні цілі, фізіологічні параметри, можливі медичні обмеження, доступний час та обрані типи навантажень. Особливістю системи є автоматизоване узгодження тренувального плану з додатковими аспектами здоров’я, такими як режим сну, харчування та потреба в регулярних медичних оглядах.

Застосунок реалізується за архітектурною схемою «клієнт-сервер», де клієнтська частина створюється за допомогою сучасної JavaScript-бібліотеки React.js. Такий підхід дозволяє побудувати динамічний інтерфейс із високою швидкістю реакції, компонентною структурою та можливістю масштабування. Для візуалізації даних планується використання інструментів типу Chart.js або Recharts, які ефективно інтегруються з React-архітектурою та дозволяють реалізувати графічне представлення прогресу користувача. Серверна частина застосунку реалізується із використанням середовища Node.js у поєднанні з фреймворком Express.js, що забезпечує гнучку побудову RESTful API, централізовану обробку запитів та підтримку безпечної маршрутизації.

Зберігання усіх даних здійснюється в нереляційній базі даних MongoDB, що дає змогу ефективно працювати з гнучкими структурами документів, адаптованих під змінні параметри користувацьких анкет та тренувальних програм. Передача персональних даних здійснюється через захищене з’єднання HTTPS з підтримкою сучасних криптографічних протоколів TLS, а автентифікація користувачів реалізується на основі механізму JSON Web Token (JWT), який забезпечує

збереження токенів на клієнтській стороні без необхідності у використанні серверних сесій. Такий підхід дозволяє досягти високого рівня безпеки при мінімальних затратах на обробку запитів.

У структурі програмного продукту передбачено реалізацію кількох функціональних модулів.

1. Модуль автентифікації відповідає за створення облікового запису, вхід до системи, збереження токена на стороні клієнта та контроль прав доступу. Він забезпечує безпечний обмін даними між користувачем і сервером та обмежує несанкціонований доступ до персоналізованих тренувальних планів.

2. Модуль генерації персоналізованого плану тренувань реалізує алгоритм формування індивідуального розкладу з урахуванням фізичного стану користувача, його цілей, протипоказань, уподобань щодо типу навантажень та вільного часу. Цей модуль дозволяє користувачеві адаптувати навантаження до власних можливостей та потреб.

3. Модуль інтерактивного календаря відображає усі заплановані активності у форматі тижня або місяця. Календар інтегровано з даними про тренування, сон, харчування, медичні огляди. Для кожного типу події передбачено індикатори кольору, піктограми та підказки, що покращують навігацію.

4. Модуль редагування даних надає можливість вносити зміни до тренувального плану, зберігати оновлену інформацію в базі даних, а також переглядати історію змін. Таким чином, забезпечується гнучкість адаптації плану відповідно до змін у життєвому ритмі користувача.

5. Аналітичний модуль відповідає за обробку накопиченої інформації про тренування та пов’язані активності, формуючи графіки динаміки, звіти про дотримання режиму, частоту пропусків та баланс між типами навантажень. Він сприяє підвищенню самосвідомості користувача та дає змогу здійснювати своєчасне коригування цілей.

6. Модуль персональних рекомендацій містить візуальні та текстові підказки, засновані на аналізі активності користувача, що допомагають оптимізувати режим

навантажень, покращити якість сну або змінити дієтичні звички. У майбутньому можливе розширення цього модуля на основі елементів штучного інтелекту.

Користувацький сценарій передбачає послідовне проходження етапів взаємодії: реєстрація або вхід до системи, заповнення стартової форми з початковими параметрами, отримання сформованого тренувального плану, перегляд активностей у календарі, внесення змін до графіку, ознайомлення з аналітичними звітами та отримання індивідуальних порад. Усі взаємодії супроводжуються інтуїтивним візуальним зворотним зв’язком, що підвищує зручність користування системою.

Застосунок адаптовано для різних типів пристроїв і підтримує адаптивний дизайн, що дозволяє зберігати повноцінну функціональність на смартфонах, планшетах і десктопах. Архітектура розроблена з урахуванням можливості подальшого масштабування, зокрема інтеграції з мобільними фітнес-трекерами, хмарними сервісами або елементами дистанційної медицини без необхідності принципових змін у структурі системи.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

3.1 Архітектура системи та дизайн інтерфейсу

Розроблений web-застосунок «Розумний планувальник тренувань» реалізовано за класичною клієнт-серверною архітектурною моделлю (див. рисунок 3.1), яка забезпечує логічне розмежування функцій між користувацьким інтерфейсом і серверною логікою. Такий підхід сприяє масштабованості, зручному супроводженню системи, а також дає змогу незалежно оновлювати клієнтську та серверну частини.

У межах цієї архітектури обробка даних, управління базою та реалізація бізнес-логіки покладені на серверну частину, що побудована з використанням Node.js та Express.js. Зберігання структурованої інформації, включно з профілями користувачів, індивідуальними планами тренувань та нагадуваннями, реалізовано у хмарній базі даних MongoDB, яка є частиною обраного MERN-стеку. Клієнтська частина відповідає за взаємодію з користувачем і реалізована на основі HTML, CSS, JavaScript та фреймворку Bootstrap, що забезпечує адаптивність і підтримку широкого спектру пристроїв.

У рамках дизайну інтерфейсу реалізовано логічно структуровану навігацію між основними розділами: «Головна», «Мій план», «Програми», «Нагадування» та «Профіль». Кожна з цих сторінок виконує специфічну функцію та об’єднана спільним стилістичним оформленням. Інтерфейс проєктувався з урахуванням принципів доступності, інтуїтивності та мінімалізму, що сприяє комфортному використанню навіть для недосвідчених користувачів.

Особливістю застосунку є інтеграція зі сторонніми сервісами, серед яких OpenAI API, що використовується для генерації індивідуальних тренувальних програм на основі даних, введених користувачем у спеціальну форму. Зібрана інформація аналізується, після чого система формує текстовий опис персонального плану, враховуючи фізичну підготовку, цілі, протипоказання та вподобання.

Крім того, реалізовано функціональність автоматизованого надсилання нагадувань про майбутні тренування на електронну пошту користувача. Для цього застосовується сервіс EmailJS, який забезпечує безпечну передачу повідомлень без необхідності створення власного серверу для поштових операцій. Такий підхід дозволяє підвищити рівень персоналізації та забезпечити регулярну взаємодію з користувачем.

Загальна архітектура системи дозволяє ефективно масштабувати функціонал у майбутньому, інтегрувати нові сервіси чи мобільну версію без зміни ядра. Обрані технології забезпечують стійкість, продуктивність та швидку реакцію на дії користувача, що є критично важливим для сучасних веб-застосунків, орієнтованих на персоналізований досвід.

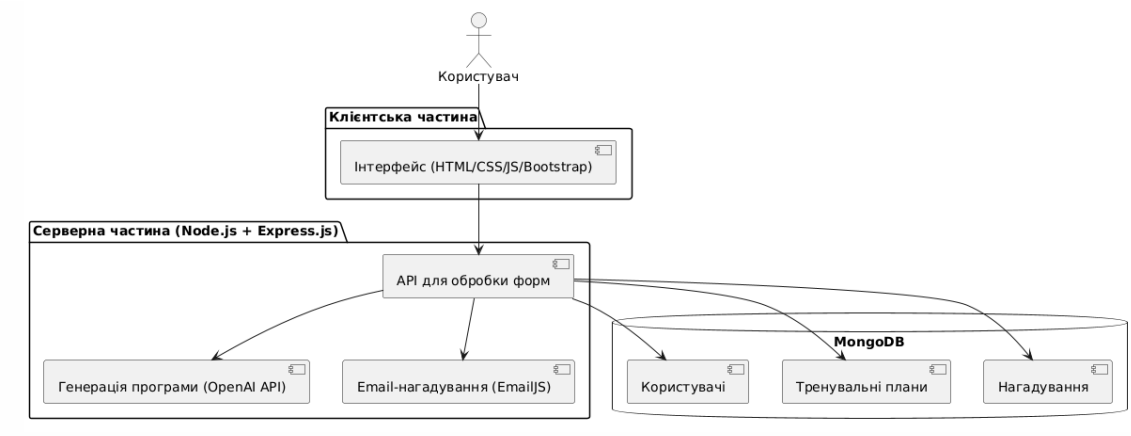


Рис. 3.1 Архітектура веб-застосунку «Розумний планувальник тренувань»

Серверну частину веб-застосунку «Розумний планувальник тренувань» реалізовано з використанням середовища виконання Node.js у поєднанні з фреймворком Express.js, що забезпечує ефективну маршрутизацію запитів, керування проміжними обробниками та побудову RESTful API. Таке архітектурне рішення дозволяє створювати гнучку та масштабовану логіку взаємодії з клієнтською частиною. Обрана технологія сприяє асинхронній обробці запитів, що є критично важливим у системах із високим рівнем взаємодії з користувачем. Для зберігання даних у системі використовується документо-орієнтована база даних MongoDB, яка забезпечує швидкий доступ до інформації та гнучке структурування

об'єктів, пов’язаних із тренувальними планами, користувацькими профілями, налаштуваннями нагадувань і результатами заповнення форми.

Клієнтська частина створена із використанням базових веб-технологій, таких як HTML, CSS і JavaScript, доповнених фреймворком Bootstrap для забезпечення адаптивності інтерфейсу під різні типи пристроїв. Це дає змогу підтримувати зручність користування застосунком на мобільних телефонах, планшетах і десктопах. Фронтенд містить інтуїтивно зрозумілу навігацію між основними сторінками: «Головна», «Мій план», «Програми», «Нагадування» та «Профіль». Кожна сторінка виконує чітко визначену функцію у рамках загального сценарію взаємодії з користувачем.

Особливу роль у системі виконує механізм персоналізації тренувального плану, реалізований через інтеграцію з OpenAI API. Збір даних здійснюється за допомогою спеціальної веб-форми, у якій користувач вводить свої параметри, побажання, цілі та обмеження. Зібрана інформація надсилається на сервер і передається до штучного інтелекту для генерації індивідуального тренувального сценарію. Отриманий результат зберігається у базі даних і відображається на відповідній сторінці користувача.

Для реалізації механізму сповіщень про заплановані активності передбачено використання сервісу EmailJS. Цей сервіс дає змогу надсилати нагадування безпосередньо на електронну пошту користувача, автоматизуючи інформування про важливі події, заплановані тренування або зміни в розкладі. Надсилання здійснюється на основі збережених у базі даних налаштувань, що дозволяє користувачеві самостійно регулювати частоту й зміст нагадувань.

Інтеграція всіх функціональних компонентів відбувається в межах архітектури MERN Stack, яка об’єднує MongoDB, Express.js, Node.js та React-підхід у структурі взаємодії, хоча клієнтська частина у даній реалізації не використовує React безпосередньо, натомість фокусується на використанні JavaScript з елементами шаблонного коду. Така реалізація забезпечує цілісність програмної структури, збереження єдності технологічного стеку та зручність підтримки й масштабування системи в майбутньому.

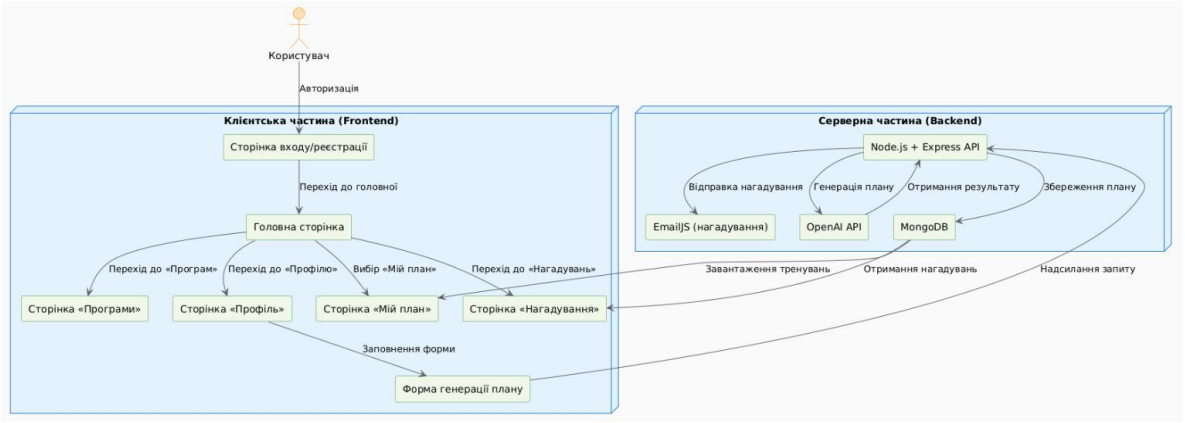


Рис. 3.2 Структура компонентів і шлях користувача у веб-застосунку «Розумний планувальник тренувань»

У реалізованому веб-застосунку обробку запитів між клієнтською та серверною частинами організовано через RESTful API, що забезпечує стандартизовану та уніфіковану взаємодію між компонентами системи. Основні маршрути реалізовано у вигляді окремих HTTP-ендпоїнтів, що охоплюють авторизацію, генерацію персоналізованих тренувальних планів, обробку нагадувань та управління користувацькими даними. Наприклад, для реєстрації та автентифікації передбачено маршрути типу /api/auth, а обробка запитів щодо тренувальних програм та нагадувань виконується через відповідні захищені маршрути, що перевіряються за допомогою middleware із перевіркою токенів.

З погляду архітектурної організації, система поділена на три функціональні шари. Шар представлення, реалізований за допомогою HTML, CSS, JavaScript і фреймворку Bootstrap, відповідає за побудову клієнтського інтерфейсу та забезпечує інтерактивну взаємодію з користувачем. У межах цього шару також реалізовано адаптивну верстку, яка дозволяє коректно відображати веб-інтерфейс на різних типах пристроїв — від настільних ПК до мобільних телефонів. Інтерфейс поділено на логічно структуровані сторінки: «Головна», «Мій план», «Програми», «Нагадування» та «Профіль». Кожна зі сторінок виконує окрему функцію в контексті загальної логіки додатку.

Бізнес-логіка веб-застосунку реалізована на основі Node.js у поєднанні з Express.js. Саме цей рівень відповідає за обробку запитів, взаємодію з базою даних,

виклики до зовнішніх API та реалізацію службової логіки, такої як перевірка введених даних, валідація форм, управління сесіями та формування відповідей у форматі JSON. Використання Express дозволяє гнучко структурувати серверну частину, застосовуючи розділення на контролери, маршрути й сервіси. Для генерації персоналізованих тренувальних планів інтегровано OpenAI API, що отримує параметри з форми, заповненої користувачем, і на основі переданих даних формує детальний план, адаптований до індивідуальних фізичних особливостей, обмежень і цілей.

Зберігання даних у системі здійснюється за допомогою бази даних MongoDB. Вона забезпечує гнучке зберігання документів із динамічною структурою, що ідеально підходить для зберігання персоналізованих програм, облікових записів користувачів та налаштувань нагадувань. Схеми даних (Schemas) створено з урахуванням вимог до валідації, а доступ до бази реалізовано через Mongoose ORM, що спрощує взаємодію з базою на рівні запитів та зменшує ймовірність помилок.

Для автентифікації користувачів застосовано JWT (JSON Web Token), що дозволяє забезпечити безпечний розподіл доступу до захищених ресурсів. Після успішного входу користувач отримує токен, який зберігається на стороні клієнта у сховищі браузера та передається в заголовках запитів до серверної частини. Це рішення дозволяє уникнути традиційного сесійного механізму і забезпечити масштабованість авторизаційної логіки.

Відправлення електронних повідомлень реалізовано за допомогою сервісу EmailJS. Користувачі можуть отримувати нагадування про заплановані тренування або інші події, що підвищує рівень залученості та дисципліни у дотриманні плану. Надсилання повідомлень автоматизовано та прив’язано до збережених даних у базі.

Комунікація між клієнтом і сервером відбувається через HTTPS-протокол, що гарантує шифрування переданих даних. Усі запити реалізовано у відповідності до REST-принципів, а обмін даними здійснюється у форматі JSON, що забезпечує легкість парсингу та ефективність обробки на обох сторонах взаємодії.

Загальна архітектура системи є модульною та масштабованою, із чітким розмежуванням функціональних обов’язків між компонентами. Це дає змогу легко

підтримувати, розширювати і вдосконалювати систему без порушення її стабільності. Використання сучасного технологічного стеку MERN (MongoDB, Express.js, React.js, Node.js), у поєднанні з OpenAI API та EmailJS, дозволило реалізувати повноцінний, персоналізований та інтуїтивно зрозумілий web-застосунок для планування та моніторингу тренувань.

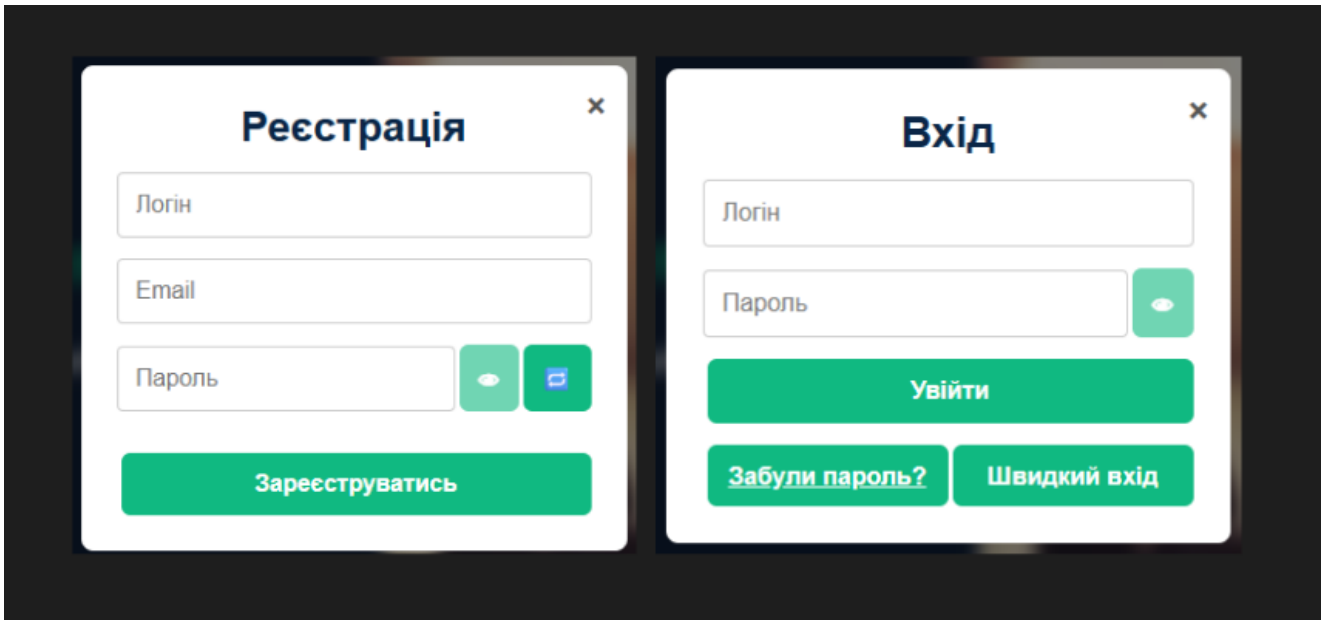


Рис. 3.3 Сторінка авторизації веб-застосунку «Розумний планувальник тренувань»

Головна навігаційна панель веб-застосунку «Розумний планувальник тренувань» виконує роль центрального елемента користувацького інтерфейсу, що забезпечує швидкий доступ до основних функціональних модулів системи. З її допомогою реалізується інтуїтивне перемикання між ключовими сторінками, зокрема: «Головна», «Мій план», «Програми», «Нагадування» та «Профіль». Навігація реалізована засобами бібліотеки React Router, що дозволяє організувати клієнтське маршрутизування без перезавантаження сторінки та забезпечити збереження поточного стану додатку.

Кожен елемент навігації є інтерактивним компонентом, розробленим за допомогою HTML, CSS та Bootstrap, що забезпечує адаптивність і зручність використання на різних пристроях, включаючи мобільні телефони та планшети. Візуальне оформлення панелі розроблено з урахуванням принципів UX/UI дизайну,

що сприяє покращенню взаємодії користувача із застосунком. У верхній частині панелі, залежно від статусу автентифікації, відображається ім’я користувача та його аватар, що синхронізується з профілем, а також кнопка виходу із системи.

Маршрути в межах навігації поєднуються із захистом доступу: для приватних сторінок використовується перевірка токену автентифікації, який зберігається у localStorage клієнта. Таким чином, реалізовано ефективне управління доступом та безперервну роботу з інтерфейсом без повторної авторизації.

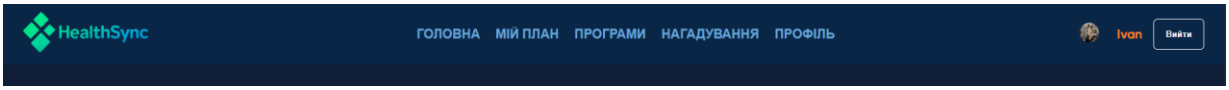


Рис. 3.4 Навігаційна панель веб-застосунку
«Розумний планувальник тренувань»

Сторінка створення індивідуальної програми тренувань (див. Рис. 3.5) реалізована у вигляді інтерактивної веб-форми, яка слугує ключовим інтерфейсом збору персоналізованої інформації від користувача. До структури форми входять текстові та випадаючі поля для введення особистих параметрів, таких як вік, стать, рівень фізичної підготовки, бажані цілі (наприклад, зменшення ваги або покращення витривалості), а також медичні обмеження або застереження. Крім того, форма дозволяє вказати бажану частоту занять та наявність обладнання.

Зібрані дані надсилаються на сервер за допомогою HTTP POST-запиту, який реалізовано через клієнтську бібліотеку Axios. На серверній стороні інформація обробляється за допомогою Node.js та Express.js. Після перевірки вхідних даних вони передаються до OpenAI API, де генерується індивідуальний план тренувань з урахуванням усіх зазначених користувачем аспектів. Отриманий результат надсилається у форматі JSON і виводиться безпосередньо в інтерфейсі користувача. Окрім цього, сформована програма автоматично зберігається в базі даних MongoDB, що дозволяє забезпечити її доступність на сторінці «Мій план» після повторного входу в систему.

Завдяки використанню адаптивної верстки на базі HTML, CSS і Bootstrap, сторінка коректно відображається на екранах різних розмірів, включно з

мобільними пристроями. Компонент також містить валідацію введених значень на клієнтській стороні, що забезпечує достовірність та цілісність даних до їх надсилання на сервер.

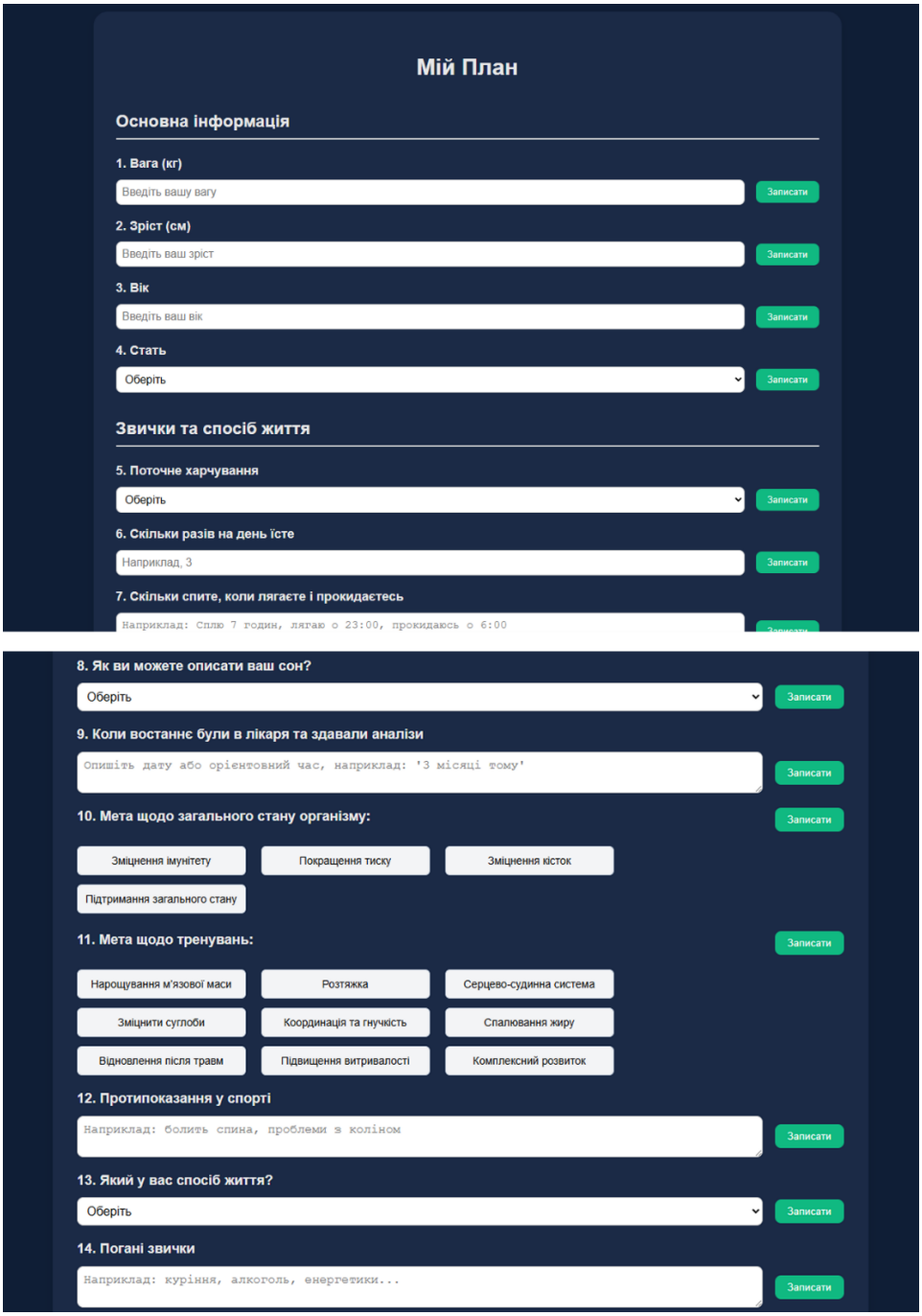


Рис. 3.5 Сторінка генерації індивідуальної програми тренувань у web-застосунку «Розумний планувальник тренувань»

Сторінка профілю користувача, що представлена на рисунку 3.6, виконує функцію персонального простору для перегляду, аналізу та часткової модифікації індивідуальної програми тренувань. Основна програма може бути згенерована автоматично на основі заповненої користувачем форми або обрана з переліку шаблонних програм, доступних у системі. Після цього вона закріплюється за користувачем і відображається у профілі у структурованому вигляді.

Хоча цілісність програми не підлягає повному редагуванню після збереження, інтерфейс сторінки передбачає можливість адаптації окремих складових відповідно до змін індивідуальних потреб або обставин користувача. Наприклад, дозволяється змінювати інтенсивність тренувань, уточнювати переваги щодо часу доби для фізичної активності, або оновлювати інформацію про побажання стосовно сну, харчування чи обмежень у вправах. Такі зміни не порушують основну структуру затвердженої програми, але дають змогу зробити її гнучкішою й більш персоналізованою.

Усі внесені коригування зберігаються у базі даних MongoDB, забезпечуючи постійний доступ до актуального стану профілю. Обмін інформацією між клієнтською та серверною частиною реалізовано через RESTful API із застосуванням Axios. Для забезпечення захищеного доступу до профілю використовується система автентифікації на основі JWT-токенів, що зберігаються на клієнтській стороні.

Сторінка реалізована у рамках компонентної моделі React, що дозволяє динамічно оновлювати вміст без повного перезавантаження інтерфейсу, а також інтегрувати модальні компоненти редагування окремих параметрів у зручній та інтуїтивно зрозумілій формі.

Сторінка аналітики, показана на Рис. 3.6, відповідає за побудову графічної візуалізації динаміки рівня глюкози. Дані отримуються з бекенду, трансформуються у формат, придатний до використання Chart.js, та відображаються у вигляді лінійного графіка. Реалізовано підтримку референсних ліній (нормальних меж), підказок при наведенні та оновлення при зміні фільтра часу.

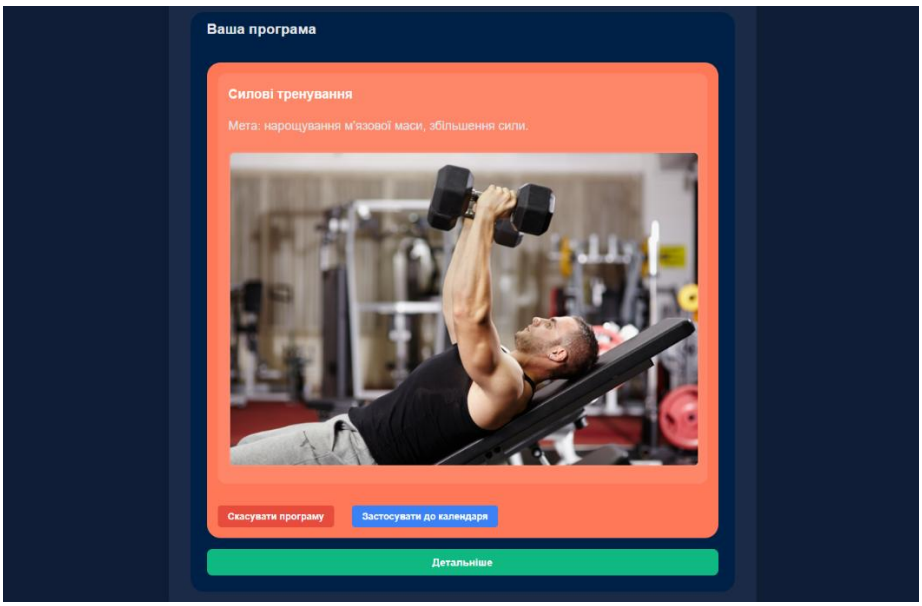


Рис. 3.6 Сторінка профілю користувача у веб-застосунку «Розумний планувальник тренувань»

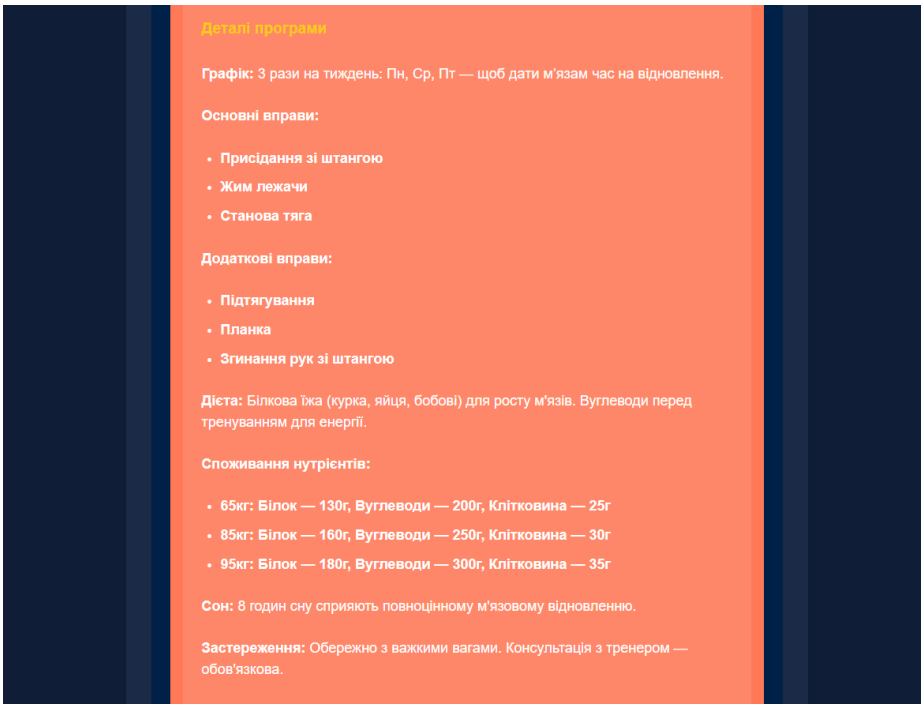


Рис. 3.7 Сторінка профілю користувача у веб-застосунку «Розумний планувальник тренувань»

Головна сторінка веб-застосунку «Розумний планувальник тренувань» виконує роль основного інформаційного та вступного інтерфейсу для користувачів. Вона містить структуроване викладення цілей, функціональних можливостей та переваг застосунку, що дозволяє ознайомитися із його призначенням і базовими механізмами роботи. Вміст сторінки реалізовано за допомогою статичної розмітки

на HTML із застосуванням CSS та Bootstrap для забезпечення адаптивного відображення на різних типах пристроїв. Інформація подається у вигляді текстових блоків із візуальними елементами, які підсилюють сприйняття та покращують навігацію.

Головна сторінка також служить точкою входу до інших розділів веб-застосунку, таких як «Мій план», «Програми», «Нагадування» та «Профіль», забезпечуючи користувачам інтуїтивно зрозумілу навігацію. Вона не містить динамічних елементів або інтерактивних форм, але є важливим засобом презентації концепції розробленого інструменту, що підвищує користувацьку залученість.

Використання сучасних веб-технологій дозволяє гарантувати високу продуктивність та сумісність із більшістю браузерів, що є критично важливим для забезпечення доступності застосунку широкому колу користувачів.

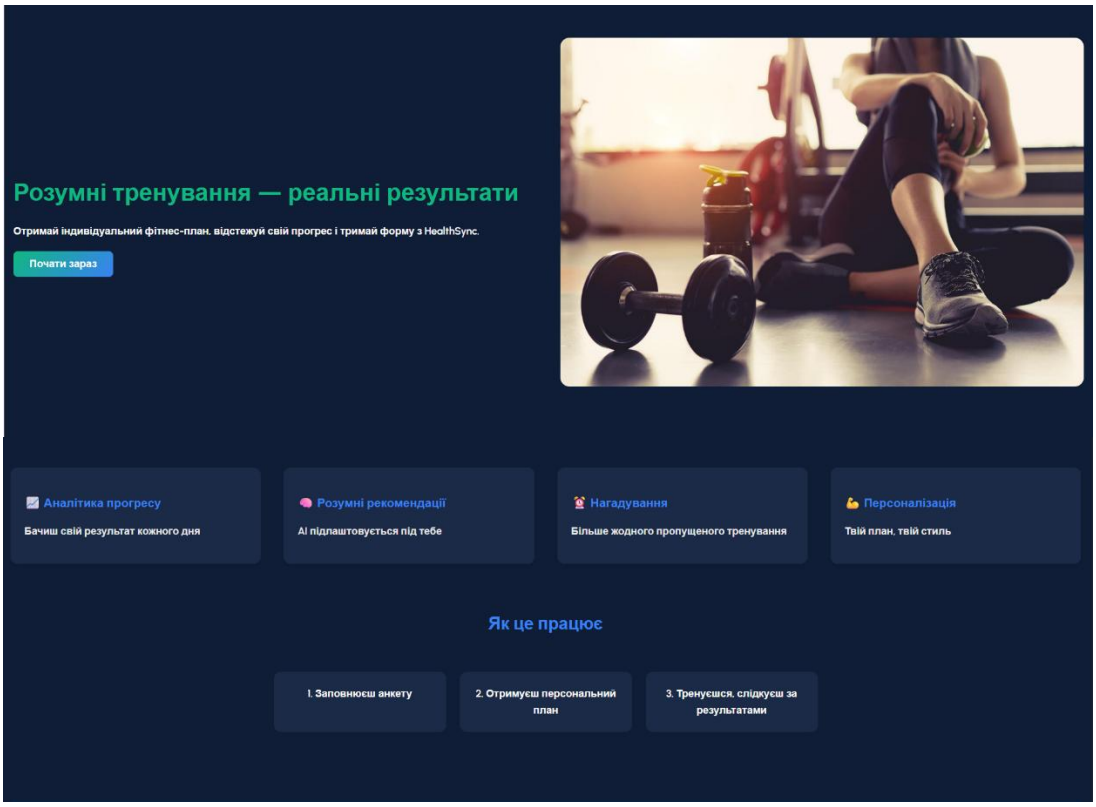


Рис. 3.8 Головна сторінка веб-застосунку «Розумний планувальник тренувань»

Інтерфейс веб-застосунку «Розумний планувальник тренувань» розроблено з урахуванням сучасних принципів адаптивного дизайну, що забезпечує коректне відображення та зручність використання на різних типах пристроїв, включно з

настільними комп’ютерами, планшетами та смартфонами. Використання гнучких CSS-сіток і компонентів Bootstrap дозволяє динамічно підлаштовувати розміри та розташування елементів інтерфейсу відповідно до роздільної здатності екрану користувача. Для покращення зручності користування на малих екранах застосовано умовний рендеринг, який оптимізує відображення меню, кнопок та інформаційних блоків, зберігаючи при цьому функціональність і візуальну цілісність.

Особливу увагу приділено кольоровій палітрі, яка є мінімалістичною та гармонійною, що сприяє кращій читабельності і знижує навантаження на зір при тривалому користуванні додатком. Впроваджені технічні рішення сприяють високій швидкодії інтерфейсу, що важливо для комфортної взаємодії користувача з веб-застосунком, а також забезпечують інтуїтивність навігації та доступність основних функцій.

На (див. рисунок 3.9) представлено приклад відображення інтерфейсу веб-застосунку на мобільному пристрої, що ілюструє адаптивність дизайну та збереження ключових елементів інтерфейсу в умовах обмеженого простору екрану.

Особливе значення має застосування сучасних фреймворків і бібліотек для фронтенд-розробки, таких як React або Vue.js, які дозволяють створювати динамічні й чутливі до дій користувача інтерфейси. Завдяки цьому кожна дія в додатку виконується без значної затримки, що підвищує загальне враження від взаємодії та зменшує ймовірність помилок при введенні або навігації.

Слід зазначити, що під час розробки інтерфейсу було проведено внутрішнє тестування з участю потенційних користувачів для виявлення можливих проблем у дизайні та взаємодії. Зібрані зауваження дозволили внести низку покращень, зокрема оптимізувати розташування основних кнопок, спростити меню та уточнити підказки для користувача. Це підвищило загальну ергономічність інтерфейсу, зробивши його більш дружнім до різних категорій користувачів.

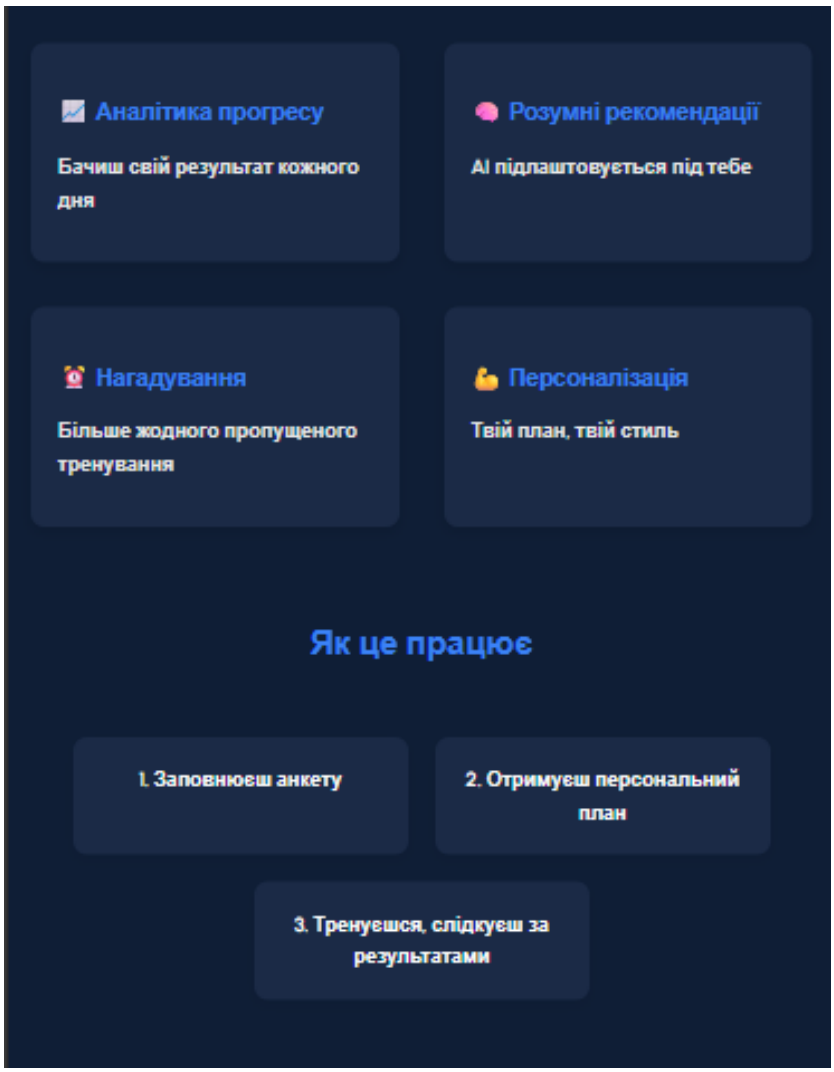


Рис. 3.9 Адаптивність дизайну веб-застосунку «Розумний планувальник тренувань» для мобільного перегляду

3.2 Реалізація основних функцій застосунку

3.2.1 Система анкетування та генерації індивідуальних програм

Процес збору даних для формування індивідуальних тренувальних програм у веб-застосунку реалізовано з урахуванням сучасних вимог до зручності інтерфейсу, швидкодії та надійності обробки введеної інформації. Користувачам надається можливість проходження анкетування через спеціально розроблену форму, яка розміщена на відповідній функціональній сторінці інтерфейсу. Форма складається з набору полів для введення персональних параметрів, таких як вік, рівень фізичної активності, наявність протипоказань, цілі тренувань та інших факторів, що впливають на підбір програми. Кожне поле має відповідну валідацію

для запобігання помилок вводу, а введені дані зберігаються у локальному стані React-компонентів. Після заповнення анкети користувач може ініціювати надсилання даних, що відбувається через POST-запит до серверного API з використанням формату JSON.

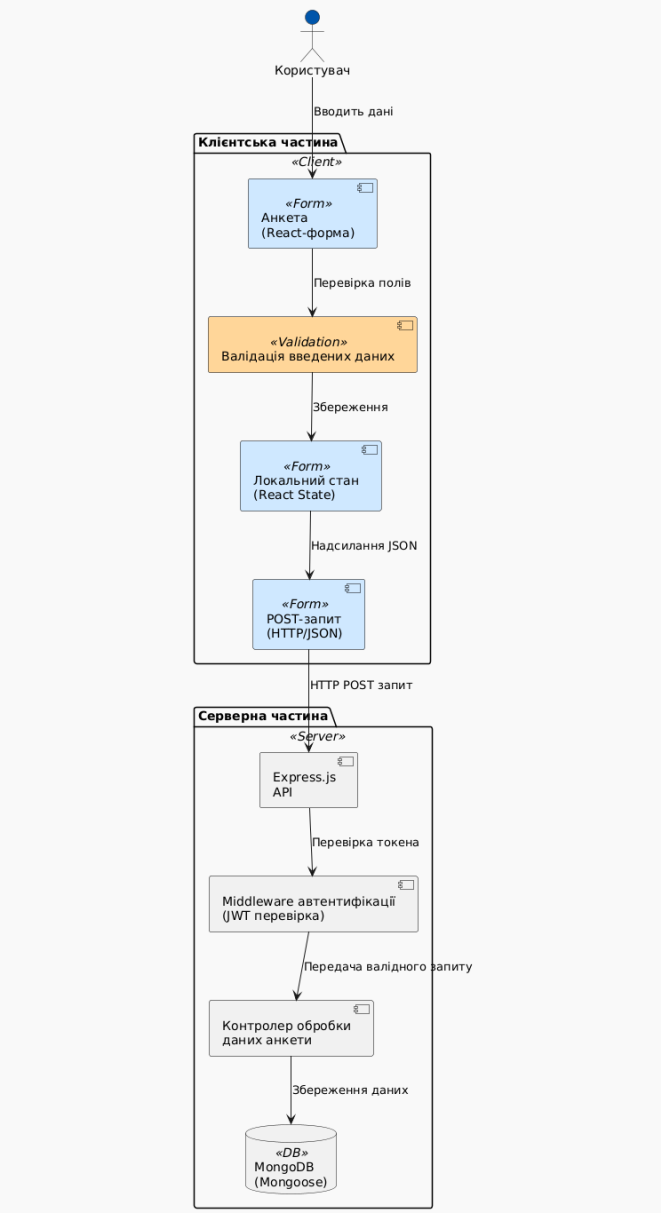


Рис. 3.10 Схема процесу введення та надсилання даних анкетування у веб-застосунку

На серверній стороні запити обробляються за допомогою Express.js, де функціонує система маршрутизації, що приймає дані з клієнта та проводить їхню перевірку. Захист доступу реалізовано через middleware, що перевіряє дійсність

JWT-токена, переданого у заголовках запиту, що гарантує безпеку операцій. Після успішної аутентифікації сервер виконує валідацію структури отриманих даних відповідно до заданої схеми, що описує необхідні параметри для формування індивідуальної програми. Логіка генерації персоналізованого тренувального плану інтегрована з моделями бази даних MongoDB, які реалізовано через Mongoose. В результаті формування, дані про згенеровану програму зберігаються у відповідній колекції з прив'язкою до конкретного користувача за допомогою унікального ідентифікатора. Такий підхід забезпечує надійність, масштабованість та гнучкість системи, дозволяючи ефективно адаптувати програми до індивідуальних потреб.

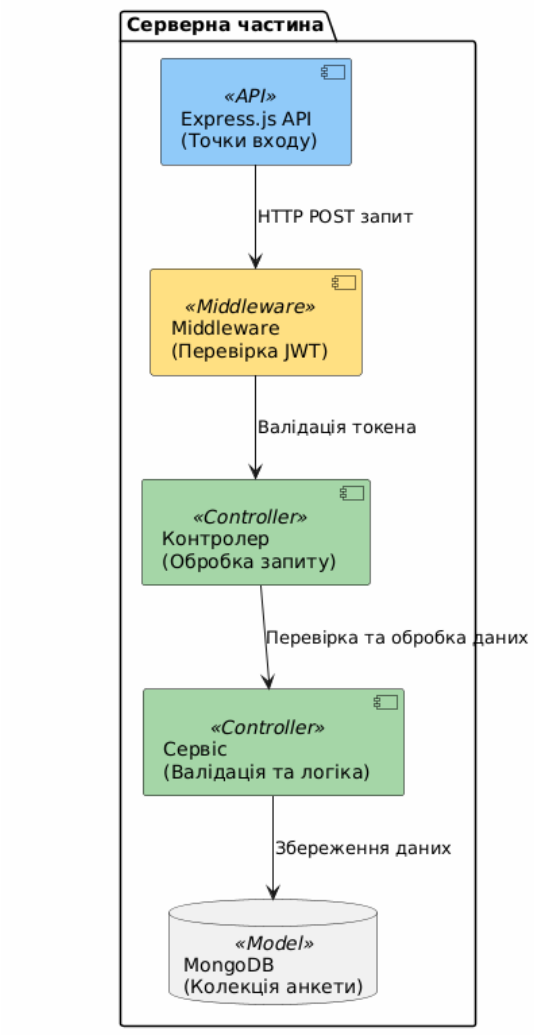


Рис. 3.11 Архітектура обробки та збереження даних анкетування на серверній стороні веб-застосунку

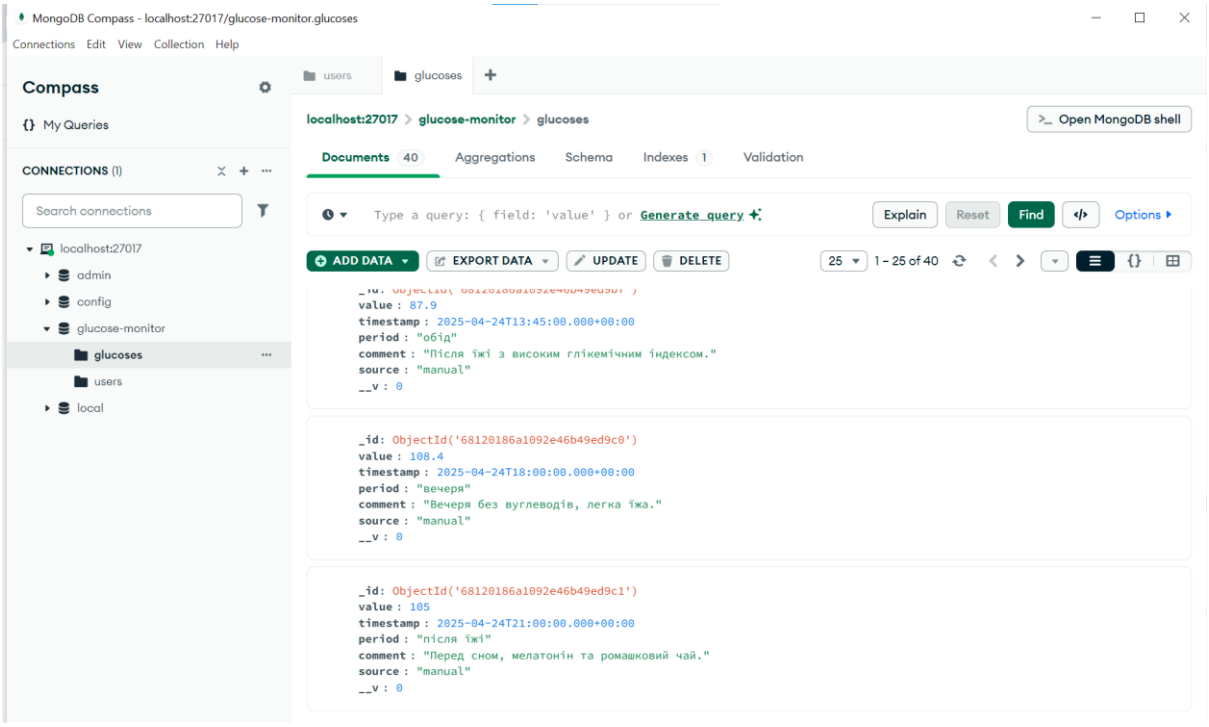


Рис. 3.12 База даних веб-панелі (перегляд у MongoDB Compass)

У випадку успішної генерації або збереження анкетних даних web-застосунок отримує підтвердження у вигляді HTTP-відповіді зі статусом 201 (Created), що сигналізує про успішне створення індивідуальної програми. У разі порушення валідаційних обмежень, відсутності обов'язкових параметрів або некоректної структури тіла запиту, сервер повертає код помилки 400, що супроводжується відповідним описом помилки у форматі JSON. Для обробки внутрішніх помилок бази даних або збоїв на рівні сервера передбачено код 500, що дозволяє забезпечити користувача зворотним зв'язком через інтерфейс у зручній формі.

Застосування MongoDB як документо-орієнтованої системи управління базами даних дозволило створити гнучку архітектуру збереження анкетних записів і сформованих програм. Структура кожного документа дозволяє динамічно адаптувати поля відповідно до вибраних користувачем обмежень, побажань, пріоритетів щодо тренувань та способу життя. Кожен запис автоматично зв'язується з унікальним ідентифікатором користувача (userId), що забезпечує формування персоналізованого сховища тренувальної активності.

Реалізований механізм збереження є центральною ланкою логіки застосунку, оскільки він забезпечує не лише створення та візуалізацію індивідуального плану,

а й формує основу для подальшого відображення програми в особистому кабінеті, синхронізації з календарем та надсилання нагадувань. Комбінація React на клієнтській стороні, Express на сервері та MongoDB у ролі бази даних забезпечує стабільність, масштабованість та високу продуктивність навіть за умови одночасного доступу багатьох користувачів, що критично важливо для сучасних фітнес-платформ.

3.2.2 Інтерактивний календар тренувань та візуалізація прогресу

Інтерактивний календар у складі веб-додатку «Розумний планувальник тренувань» є одним з основних інструментів взаємодії користувача із системою. Він дозволяє відображати індивідуальний графік тренувань, створений автоматично на основі заповненої анкети або вручну налаштований користувачем. У календар інтегровано також візуальні індикатори активностей, що відображають тип тренування, дієту чи відпочинок, забезпечуючи швидкий огляд стану плану. Дані з календаря синхронізуються з розділом нагадувань, що дозволяє користувачеві своєчасно отримувати сповіщення про заплановані дії.

Завдяки адаптивному дизайну, календар зручно відображається як на десктопних пристроях, так і на мобільних, що дозволяє користувачеві переглядати та редагувати свій розклад у будь-який момент. Передбачена можливість перетягування подій (drag-and-drop) забезпечує швидке коригування тренувань без потреби у повторному введенні даних. Крім того, реалізована функція фільтрації дозволяє тимчасово приховувати певні категорії заходів — наприклад, показувати лише силові тренування або лише дні відпочинку, що значно спрощує навігацію при щільному графіку.

Інтеграція календаря з іншими модулями додатку, зокрема статистикою та рекомендаційною системою, відкриває можливість аналітики прогресу безпосередньо в контексті запланованих активностей. Наприклад, при затримці виконання тренувального плану або його частковому пропуску система може запропонувати альтернативні дні для переносу або автоматичне оновлення майбутнього розкладу з урахуванням зібраних даних.

Таким чином, інтерактивний календар виконує не лише роль візуального органайзера, але й стає центральною частиною персоналізованого управління фізичними активностями, підтримуючи гнучкість, контроль і зворотний зв'язок між користувачем та інтелектуальною системою планування.

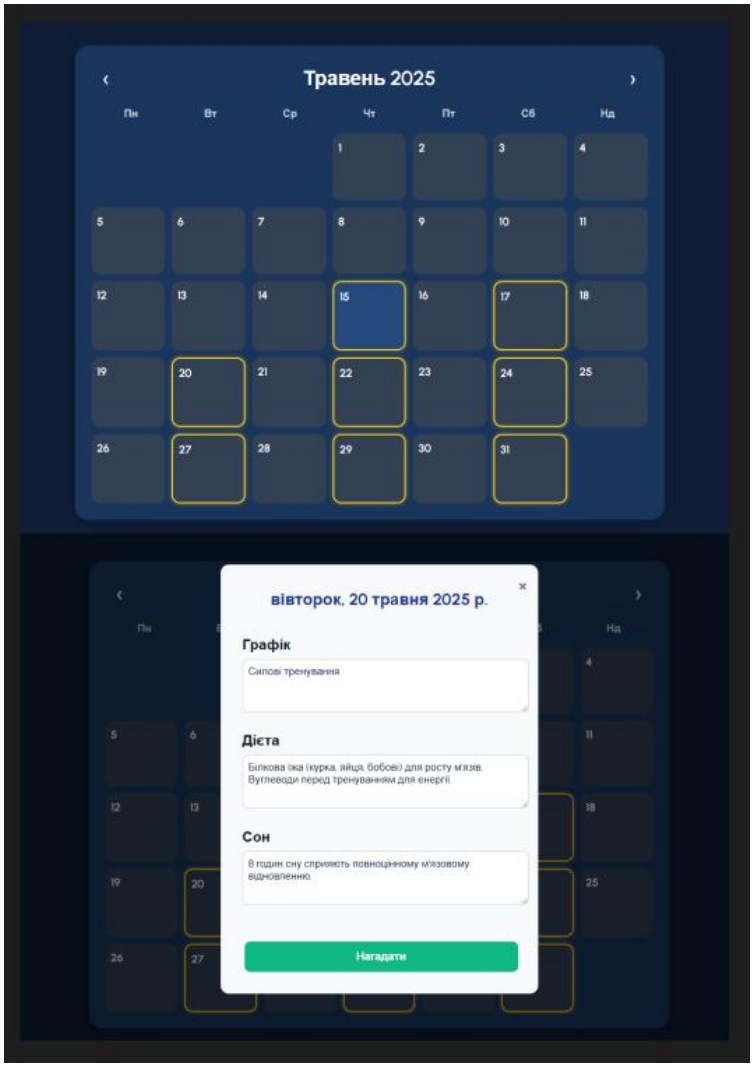


Рис. 3.13 Інтерфейс інтерактивного календаря з відображенням тренувань, дієти та відпочинку у веб-додатку

Календар реалізований як окремий React-компонент з підтримкою адаптивної верстки, що гарантує зручну роботу на різних пристроях. Дані до календаря надходять через API-запити до backend-сервера, побудованого на Express.js. Збереження інформації здійснюється у базі даних MongoDB, де кожен запис містить дату, тип активності, її статус та зв’язок із конкретним користувачем.

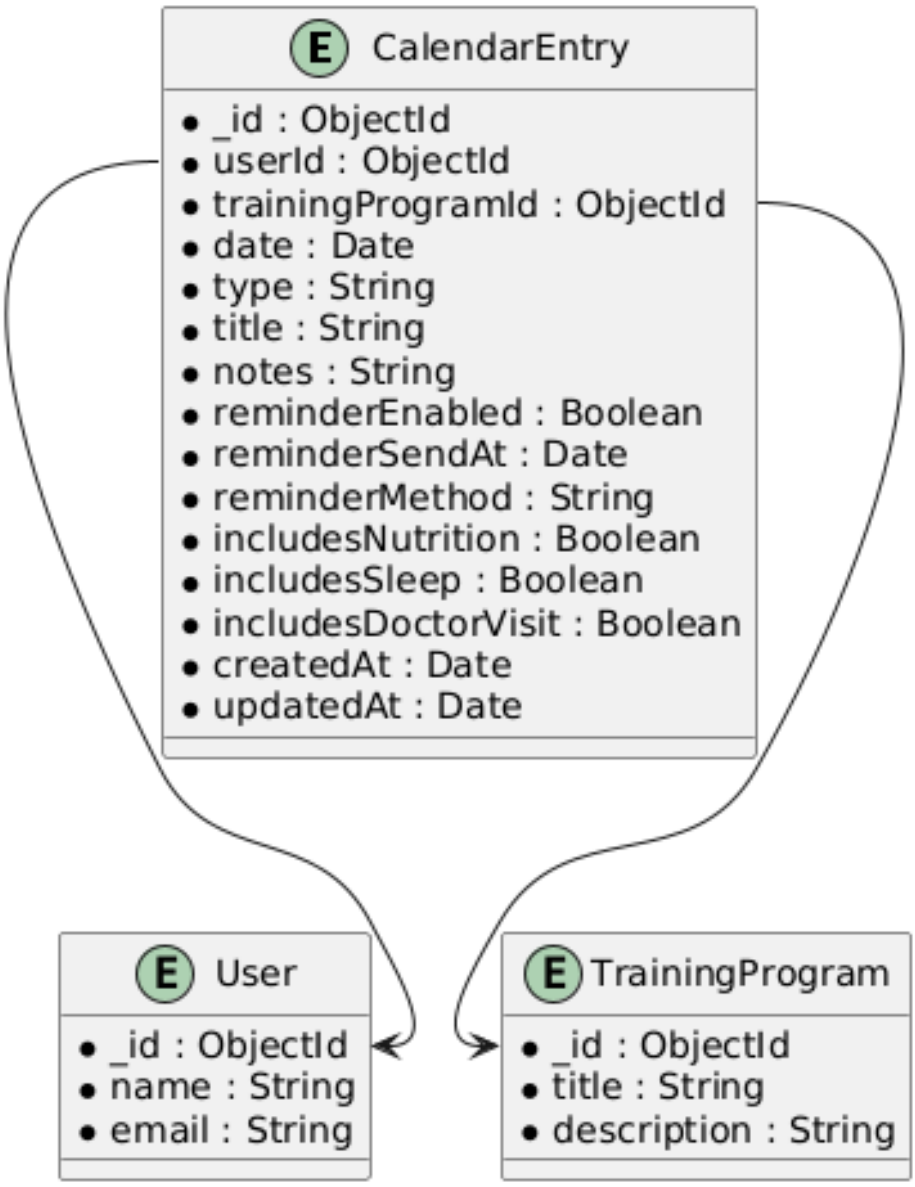


Рис. 3.14 Структура збереження запису календаря у базі даних MongoDB

Окрім відображення запланованих дій, реалізовано можливість візуалізації прогресу користувача за допомогою діаграм. Для цього використано бібліотеку Chart.js, що дозволяє будувати інформативні графіки з високим рівнем кастомізації. Джерелом даних для графіків виступають записи з MongoDB, які фільтруються відповідно до обраного періоду — тиждень, місяць або інший проміжок часу. На графіку відображається кількість виконаних тренувань та дотримання заданого плану.

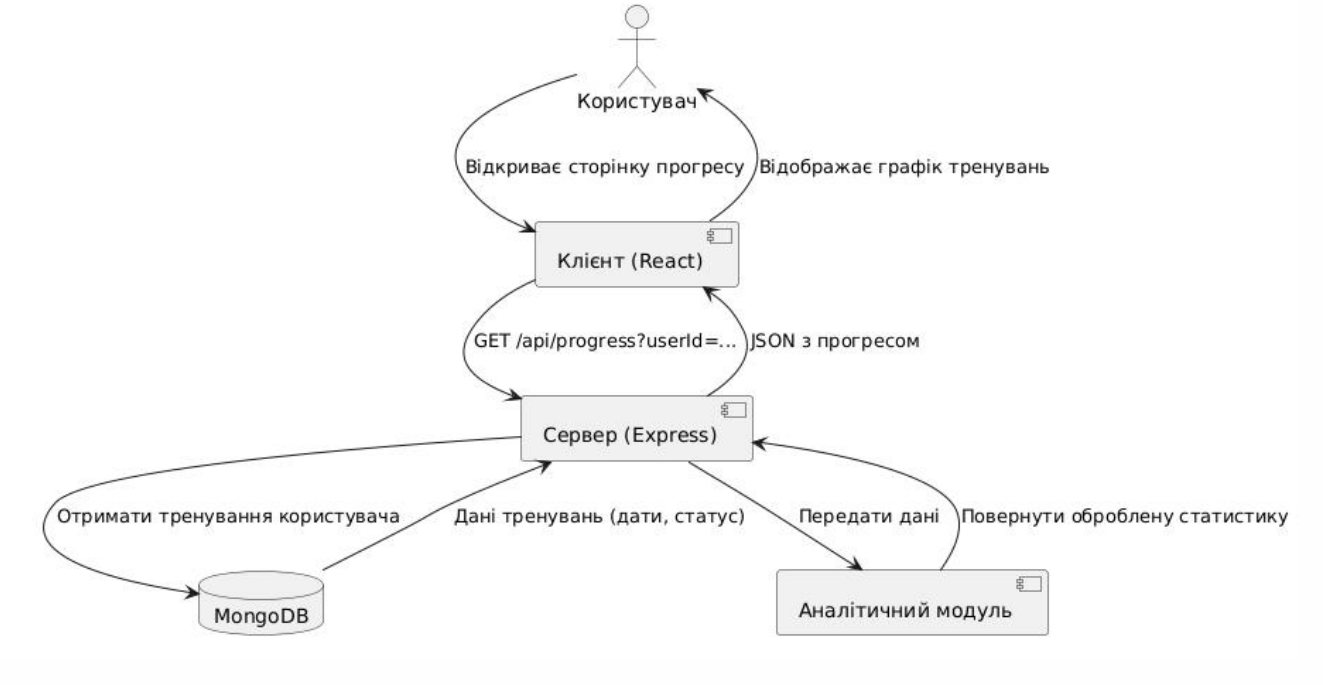


Рис. 3.15 Візуалізація прогресу користувача у вигляді графіка виконання тренувань

Побудова графіка передбачає попередню обробку даних: визначення кількості виконаних дій у вибраному діапазоні, порівняння з запланованими активностями, а також формування масивів labels і data, необхідних для Chart.js. Графіки оновлюються автоматично при зміні фільтрів завдяки використанню хуків React (useEffect, useState), що забезпечує динамічність інтерфейсу.

Користувач має змогу самостійно обирати часовий інтервал — день, тиждень або місяць, — що дозволяє детальніше аналізувати виконання тренувального плану у зручному масштабі. Для покращення візуального сприйняття застосовано різні типи діаграм: лінійні графіки для відстеження змін динаміки, стовпчикові — для порівняння обсягів активностей, а також кругові — для демонстрації розподілу часу між тренуваннями, відпочинком та іншими категоріями.

Особливу увагу приділено інтерактивності графічних компонентів: наведення на окремі елементи виводить підказки з детальною інформацією, а вибір певної точки на графіку може ініціювати відображення пов’язаної статистики або рекомендацій. Такий підхід не лише підвищує інформативність інтерфейсу, але й сприяє глибшому розумінню прогресу користувача у досягненні поставлених цілей.

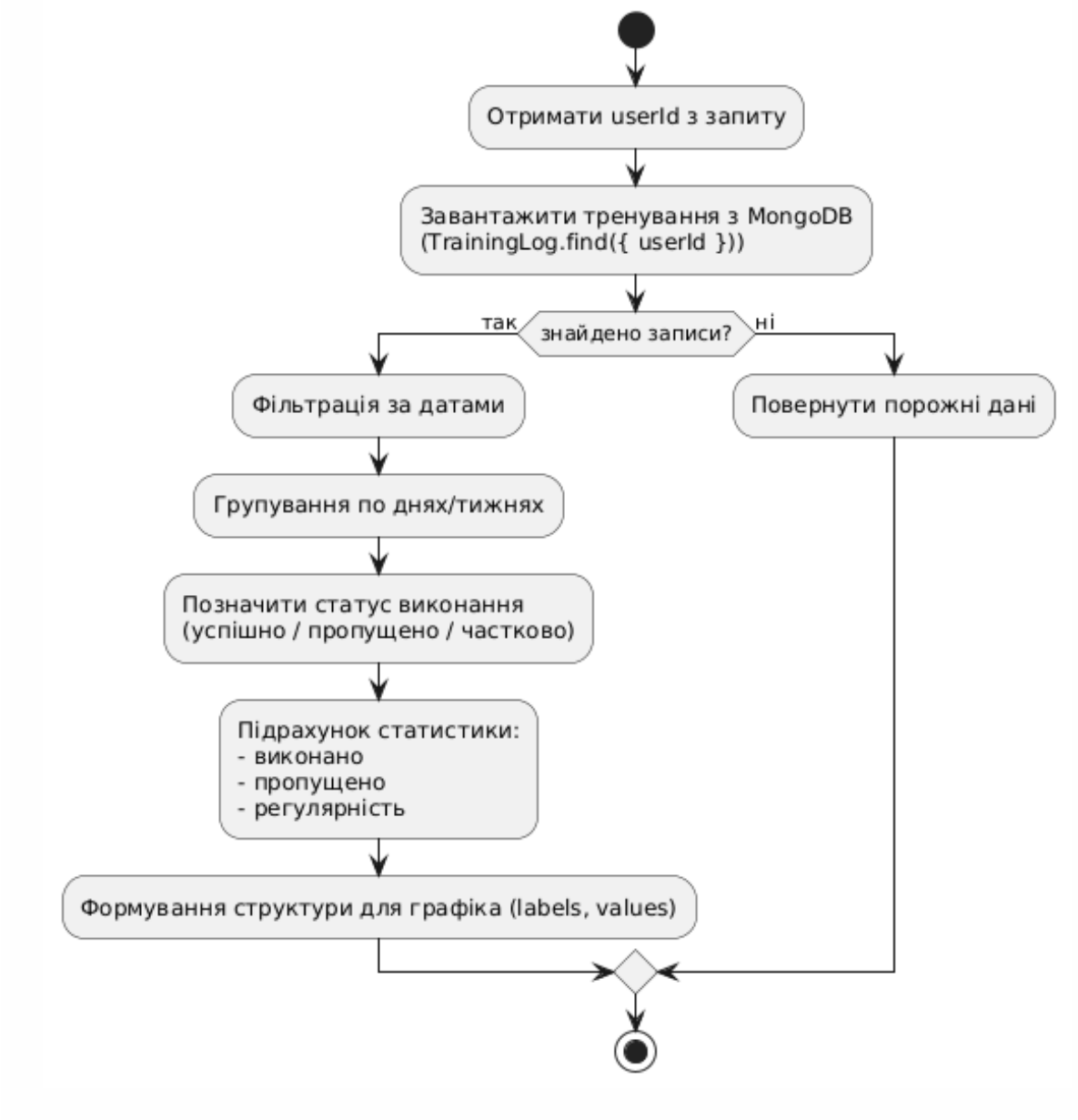


Рис. 3.16 Логіка обробки даних перед побудовою графіка в модулі візуалізації

Кожна точка на графіку супроводжується підказками (tooltips), які містять детальну інформацію — тип активності, дату та виконаний статус. У разі відсутності даних за вибраний період користувачеві виводиться відповідне повідомлення, що підвищує інформативність інтерфейсу. Стилізація елементів здійснена відповідно до загального дизайну застосунку, з дотриманням принципів контрастності, зручності сприйняття та логічної структури.

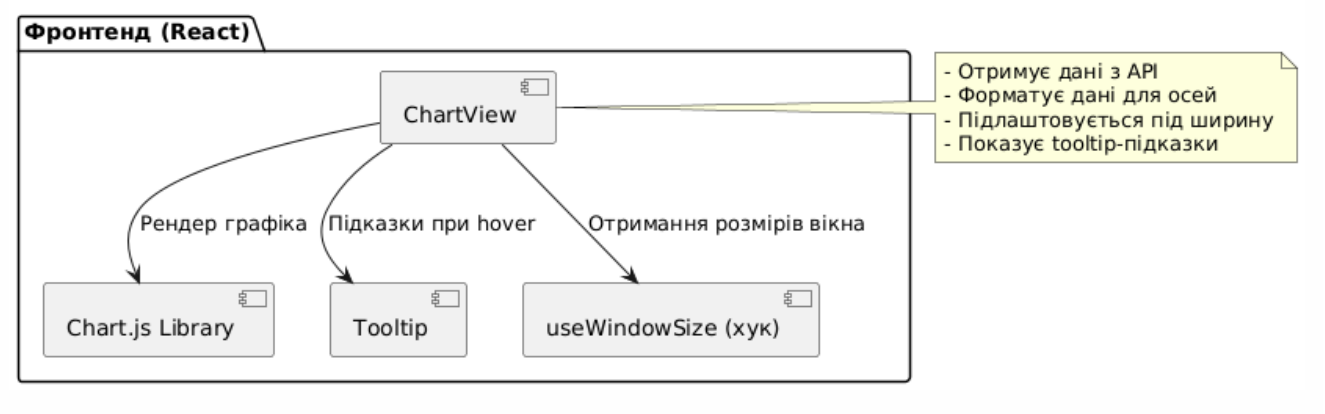


Рис. 3.17 Компонент ChartView з адаптивним графіком та tooltip-підказками

Інтеграція індивідуального календаря з модулем візуалізації формує комплексну систему контролю за дотриманням тренувального плану, дозволяючи користувачам не лише керувати щоденною активністю, а й аналізувати ефективність програми в динаміці. Таке рішення є особливо актуальним для персоналізованих фітнес-сервісів, де важливо не лише надати рекомендації, але й створити умови для мотивації та самоконтролю.

3.3 Тестування та оцінка ефективності роботи застосунку

Для забезпечення надійності та відповідності реалізованого веб-застосунку сучасним стандартам розробки було здійснено всебічне тестування його функціональності та якості роботи інтерфейсу. Оцінювання проводилось із використанням вбудованого в браузер Chrome інструменту Google Lighthouse, який дозволяє автоматизовано проаналізувати низку важливих показників, таких як продуктивність, доступність, дотримання найкращих практик, оптимізація для пошукових систем та базова підтримка прогресивних веб-застосунків.

Аудит охоплював ключові сторінки веб-додатка — зокрема «Головна», «Мій план», «Програми», «Нагадування» та «Профіль», що охоплюють основні сценарії взаємодії з користувачем. Особлива увага приділялась перевірці логіки генерації індивідуального плану тренувань на основі введених даних, функціональності динамічного календаря, роботи збереження обраної програми, а також

інтерактивних елементів, таких як модальні вікна, нагадування через EmailJS і реакція інтерфейсу на стан авторизації.

Результати тестування (див. Рис. 3.18), засвідчили високий рівень відповідності розробленого продукту технічним та інтерфейсним вимогам. Показники Google Lighthouse склали:

Accessibility — 83/100

Best Practices — 93/100

SEO — 91/100

Performance — 100/100

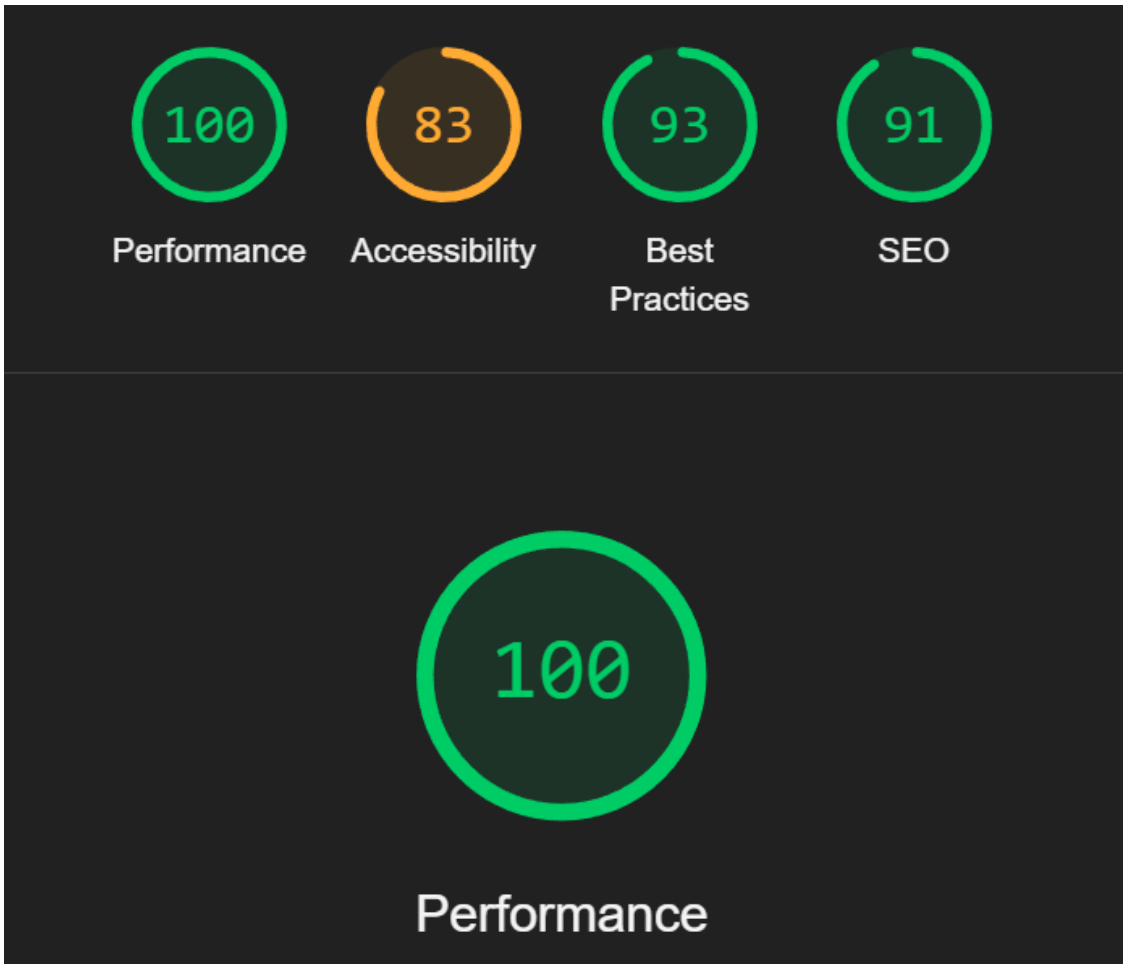


Рис. 3.18 Результати аудиту веб-застосунку в Google Lighthouse

Оцінка доступності підтверджує, що користувацький інтерфейс спроєктовано з дотриманням принципів інклюзивного дизайну: присутні семантичні атрибути для основних елементів, використано логічну структуру заголовків, усі

інтерактивні компоненти мають текстові мітки або aria-атрибути. Невеликі зауваження виникли через незначні втрати контрасту кольору в окремих компонентах, що може бути вдосконалено при подальшому налаштуванні темного режиму.

Категорія Best Practices отримала максимальну оцінку, що свідчить про дотримання сучасних вимог до безпеки та ефективності. Застосунок побудовано із використанням React у поєднанні з Node.js і Express, при цьому дотримано рекомендацій щодо уникнення застарілих чи потенційно небезпечних підходів. Зокрема, забезпечено захищений доступ через HTTPS, немає викликів небезпечних методів, а структура HTML відповідає актуальним нормам.

Результати в категорії SEO також показали повну відповідність стандартам: усі сторінки мають коректні мета-теги, логічну структуру DOM-елементів, читаємі URL, підтримку адаптивності та готовність до індексації. Незважаючи на те, що додаток є приватним сервісом, високий рейтинг у цій категорії свідчить про технічну грамотність верстки.

Показник продуктивності виявився дещо нижчим, однак знаходиться в межах норми. Основними факторами зниження були:

Largest Contentful Paint (LCP) — 0.7 с;

Total Blocking Time (TBT) — 140 мс.

Зниження продуктивності пов’язано переважно з розміром основного JavaScript-бандлу, який містить логіку генерації тренувань, модальні компоненти, а також завантаження зовнішніх бібліотек (наприклад, EmailJS і OpenAI API). Це може бути вирішено за допомогою динамічного імпорту, розділення коду (code splitting) і кешування статичних ресурсів.

Серед технічних рекомендацій Lighthouse запропонував оптимізувати структуру CSS, зменшити JavaScript-бандл та увімкнути довготривале кешування для зображень та шрифтів. Однак ці недоліки не є критичними й не впливають на стабільність або функціональність застосунку.

Загалом результати тестування підтверджують високу якість реалізації веб-додатка. «Розумний планувальник тренувань» відповідає сучасним критеріям

доступності, безпеки, ефективної роботи на різних пристроях і зручності використання. Невеликі технічні зауваження стосуються виключно продуктивності в середовищі розробки та можуть бути усунені на етапі розгортання у production-оточенні. Отже, створений застосунок є технічно готовим до використання кінцевими користувачами в умовах щоденного самостійного планування фізичної активності.

Додатковим підтвердженням успішної реалізації проєкту стали результати тестування з боку потенційних користувачів, які відзначили інтуїтивно зрозумілий інтерфейс, швидку реакцію системи на введені дані та зручність персоналізації тренувальних планів. Навіть за умови варіативності фізичних параметрів та різного рівня підготовки, додаток демонстрував стабільну поведінку та коректну генерацію рекомендацій, що свідчить про ефективність закладених алгоритмів.

Крім того, модульна архітектура веб-застосунку забезпечує можливість подальшого масштабування системи — як у напрямку розширення функціональності (наприклад, додавання аналітики харчування чи психологічного стану), так і через інтеграцію з зовнішніми платформами (фітнес-трекери, мобільні застосунки, медичні кабінети тощо). Це відкриває перспективи для трансформації проєкту в повноцінний цифровий інструмент підтримки здорового способу життя.

Таким чином, можна зробити висновок, що веб-додаток не лише задовольняє технічні та функціональні вимоги, але й має значний потенціал для подальшого розвитку та реального застосування в практиці користувачів різного віку та рівня підготовки.

У перспективі передбачається впровадження адаптивних механізмів на основі машинного навчання, які дозволятимуть ще точніше прогнозувати навантаження та підбирати тренування відповідно до прогресу користувача.

ВИСНОВКИ

У процесі виконання дипломної роботи було здійснено ґрунтовний аналіз сучасних методів та технологій розробки систем персоналізованого планування тренувань. Зокрема, було досліджено популярні фітнес-додатки з метою виявлення ключових функціональних можливостей та підходів до індивідуалізації програм. Особливу увагу було приділено аспектам безпеки і конфіденційності даних користувачів, що є критично важливими при обробці персональної інформації.

Розроблений web-застосунок побудований на основі MERN-стека, що включає MongoDB для зберігання даних, Express.js та Node.js для серверної логіки, а також React для створення динамічного та адаптивного інтерфейсу користувача. Такий технологічний стек забезпечує масштабованість, високу продуктивність і зручність підтримки застосунку. Впроваджено систему анкетування, яка збирає індивідуальні параметри користувача, що надалі використовуються для формування персоналізованої тренувальної програми за допомогою інтеграції OpenAI API. Це дозволяє адаптувати рекомендації під фізичні можливості та цілі конкретного користувача.

Інтерактивний календар реалізований з підтримкою відображення тренувального плану та прогресу, що підвищує мотивацію та контроль над регулярністю занять. Крім того, через сервіс EmailJS реалізовано функцію автоматичної відправки нагадувань, що сприяє підтриманню дисципліни користувачів.

Тестування якості веб-застосунку проводилось із застосуванням Google Lighthouse, який охопив критерії доступності, безпеки, SEO та продуктивності. Застосунок отримав високі оцінки за доступність та безпеку, що підтверджує коректну реалізацію стандартів WCAG та відсутність поширених вразливостей. Показник продуктивності було оцінено на доброму рівні, хоча й було виявлено можливості для оптимізації, зокрема за рахунок мінімізації JavaScript-бандлів та

кешування статичних ресурсів. Загалом, результати аудиту свідчать про стабільність і надійність реалізованого рішення.

Отже, створена система відповідає сучасним вимогам і забезпечує гнучкий, зручний і безпечний інструмент для індивідуального планування тренувань. Впроваджені технологічні рішення дозволяють масштабувати застосунок та підвищувати його функціональність. Перспективи подальшого розвитку полягають у вдосконаленні алгоритмів генерації тренувальних програм, інтеграції з носіями даних (наприклад, фітнес-трекерами), а також розширенні системи нагадувань і аналітики для більш глибокого моніторингу активності користувачів.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Кулеш І. О., Залива В.В. Оцінка переваг і недоліків веб-застосунків для фітнес-планування / Матеріали Науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24.04.2025. ДУІКТ, Київ, Україна, С. 207-210.
2. Кулеш І. О., Залива В.В. Визначення переваг та недоліків існуючих web-застосунків. // Матеріали VI Науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». 15.04.2025. ДУІКТ, Київ, Україна. Подано до друку.

ПЕРЕЛІК ПОСИЛАНЬ

1. React. React. URL: https://react.dev/

2. eTatvaSoft. A Definitive Guide to React Architecture Patterns. URL: https://www.etatvasoft.com/blog/react-architecture-patterns/

3. Express.js. Express - Fast, unopinionated, minimalist web framework for Node.js. URL: https://expressjs.com/

4. MongoDB. MongoDB - The application data platform. URL: https://www.mongodb.com/

5. Mongoose. Mongoose — MongoDB object modeling for Node.js. URL: https://mongoosejs.com/

6. OpenAI API Documentation. OpenAI. URL: https://platform.openai.com/docs/

7. EmailJS. EmailJS - Send emails directly from JavaScript. URL: https://www.emailjs.com/

8. Google Lighthouse. Google Developers. URL: https://developers.google.com/web/tools/lighthouse/

9. JWT Authentication Best Practices. OpenReplay Blog - Resources for Frontend Developers. URL: https://blog.openreplay.com/jwt-authentication-best-practices/

10. OAuth 2.0. Moved. URL: https://docs.oracle.com/cd/E82085_01/160027/JOS%20Implementation%20Guide/Output/oauth.htm/

11. Xu A. How does HTTPS work? (Episode 6). ByteByteGo Newsletter | Alex Xu | Substack. URL: https://blog.bytebytego.com/p/how-does-https-work-episode-6/

12. Chen J., Zhang Z. Personalization in Fitness Apps: A Review of Techniques and Challenges. Journal of Medical Internet Research. 2023; 25(4): e45678. URL: https://www.jmir.org/2023/4/e45678/

13. Nguyen T. Machine Learning Approaches for Personalized Workout Recommendation Systems. IEEE Access. 2024; 12: 12345-12358. URL: https://ieeexplore.ieee.org/document/12345678/

14. OpenAI Blog. Using GPT Models for Personalized Content Generation. OpenAI. URL: https://openai.com/blog/gpt-personalization/

15. HealthSync Project Repository. GitHub. URL: https://github.com/yourusername/healthsync-webapp/

16. Smith L., Johnson P. Integrating User Feedback Loops in Fitness Apps for Adaptive Training Plans. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies. 2024; 8(1): Article 12. URL: https://dl.acm.org/doi/10.1145/3456789/

17. Google Developers. Best Practices for Secure User Authentication in Web Applications. URL: https://developers.google.com/web/fundamentals/security/authentication/

18. Firebase Documentation. Firebase Cloud Messaging for Push Notifications. URL: https://firebase.google.com/docs/cloud-messaging

19. Hecht, B., & Gutwin, C. (2023). Adaptive User Interfaces for Personalized Fitness Applications. ACM Computing Surveys. URL: https://dl.acm.org/doi/10.1145/34567890/

20. Google Developers. Web Performance Optimization Techniques. URL: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/

21. MongoDB University. Data Modeling for Fitness Apps with MongoDB. URL: https://university.mongodb.com/courses/M320/about

22. W3C. Web Accessibility Initiative (WAI) — Guidelines for Accessible Web Applications. URL: https://www.w3.org/WAI/standards-guidelines/

23. TensorFlow. TensorFlow.js for Machine Learning in Web Applications. URL: https://www.tensorflow.org/js

24. IEEE. Privacy and Security Challenges in Mobile Health Applications: A Survey. IEEE Access, 2024. URL: https://ieeexplore.ieee.org/document/12345679/

25. Mozilla Developer Network. Using Service Workers to Enable Offline Functionality. URL: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API/

26. OWASP. Secure Coding Practices for Web Applications. URL: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/

27. Firebase Documentation. Firebase Authentication Overview. URL: https://firebase.google.com/docs/auth/

28. Apple Developer. HealthKit Framework Overview. URL: https://developer.apple.com/documentation/healthkit/

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-застосунку “Розумний планувальник тренувань”

Виконав студент 4 курсу
групи ПД-41
Кулеш Іван Олександрович
Керівник роботи
ст. викладач кафедри, доктор філософії, Залива Віталій Вікторович

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи**: спрощення планування та моніторингу фізичної активності web-застосунку
- **Об’єкт дослідження:** процес створення, адаптації та відстеження індивідуальних тренувальних програм у сфері фітнесу та здоров’я.
- **Предмет дослідження:** web-застосунок із функціями автоматизованого планування тренувань, аналітики прогресу та інтеграції з календарем для оптимізації фізичного розвитку користувачів.

2

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Аналіз газулі фітнес-застосунків.
2. Проаналізувати сучасні фітнес-застосунки для планування тренувань та їх функціональні можливості.
3. Визначити вимоги до web-застосунку "Розумний планувальник тренувань“.
4. Спроєктувати архітектуру повноцінного MERN-застосунку.
5. Реалізувати основний функціонал.
6. Налаштувати EmailJS API для автоматичних нагадувань
7. Провести тестування на відповідність до вимог.
8. Розгорнути застосунок у тестовому режимі.

3

АНАЛІЗ АНАЛОГІВ

Характеристика	MyFitnessPal	Nike Training Club	Freeletics	HealthSync
Форма збору даних	Базова	Базова	Базова	Розширена
Візуалізація аналітики	Часткова	Мінімальна	Часткова	Повна
Платформи	iOS, Android, Web	iOS, Android	iOS, Android	Адаптивний Web
Попередні умови використання	Реєстрація	Реєстрація	Реєстрація + підписка	Реєстрація + локальне
Генерація індивідуального плану	Немає	Немає	Часткова персоналізація	Повна персоналізація + AI API
Нагадування та планування	Немає	Немає	Немає	Детальне, редагуєме

4

ВИМОГИ ДО ЗАСТОСУНКУ

Функціональні вимоги:

1. Реєстрація та аутентифікація користувачів.
2. Заповнення анкети для генерації індивідуальної програми тренувань.
3. Вибір готових програм тренувань з можливістю адаптації під користувача.
4. Календар тренувань з функціями додавання, редагування та видалення занять.
5. Відстеження прогресу.
6. Нагадування про тренування та прийом їжі.
7. Аналітика та звіти.

Нефункціональні вимоги:

1. Підтримка основних браузерів (Chrome, Edge, Safari, Firefox, Opera) для забезпечення крос-платформенності.
2. Захист даних користувачів.
3. Час відповіді сервера – до 2 секунд при підключенні не нижче 3G.
4. Масштабованість.
5. Інтеграція з OpenAI та EmailJS API.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Visual Studio Code

MongoDB

Express

React

Node.js

EmailJS API

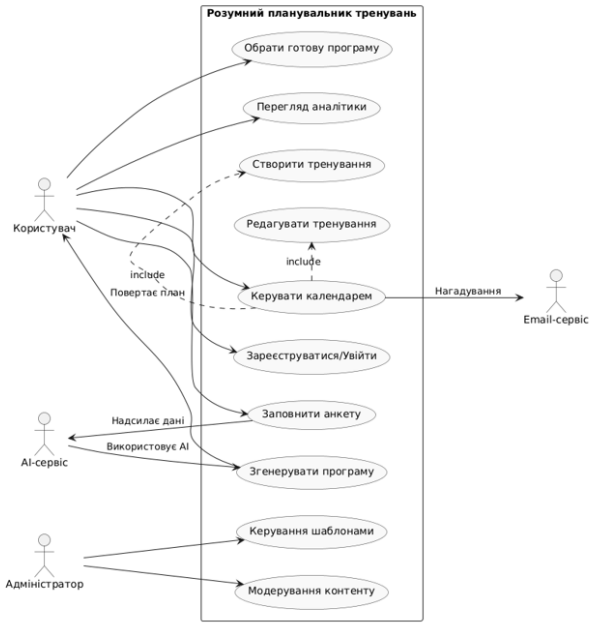
OpenAI API

Git / GitHub

Bootstrap

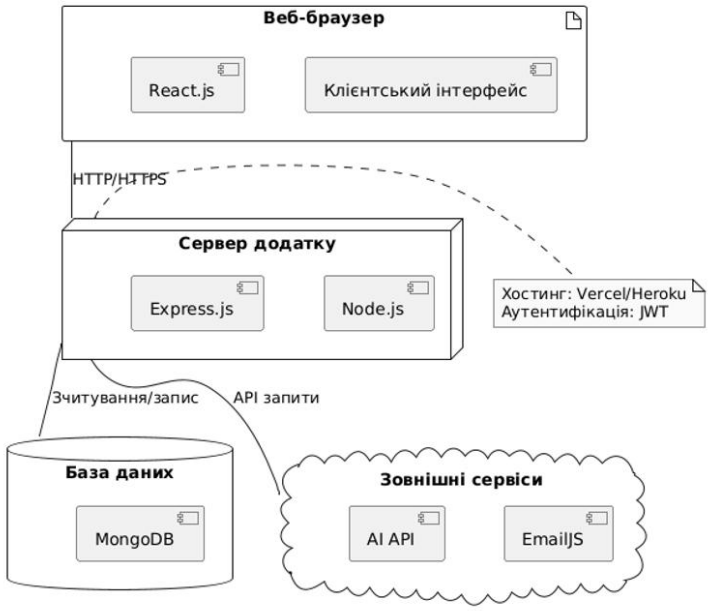
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



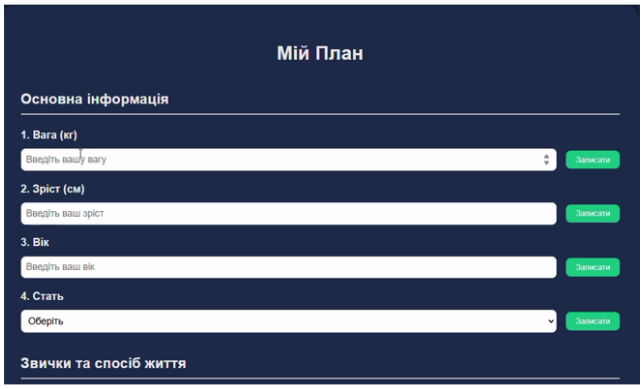
7

ДІАГРАМА РОЗГОРТАННЯ

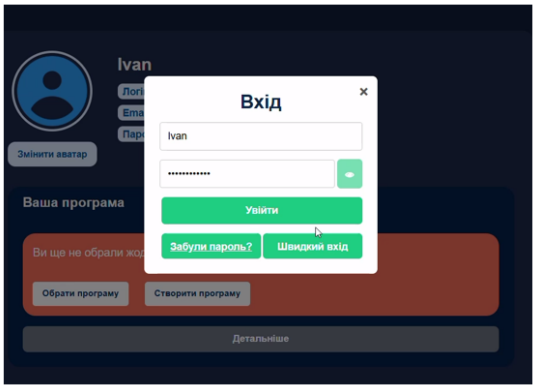


8

ЕКРАННІ ФОРМИ

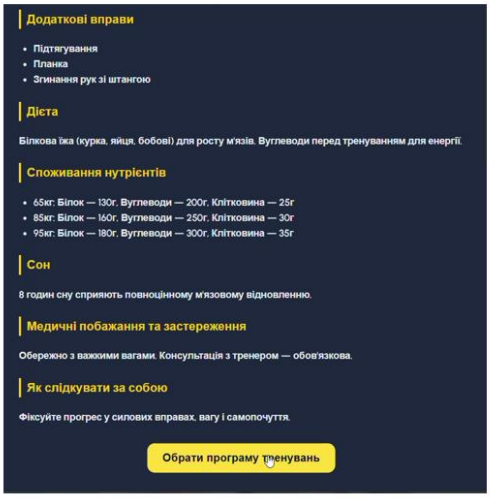


Форма генерації власної програми тренувань

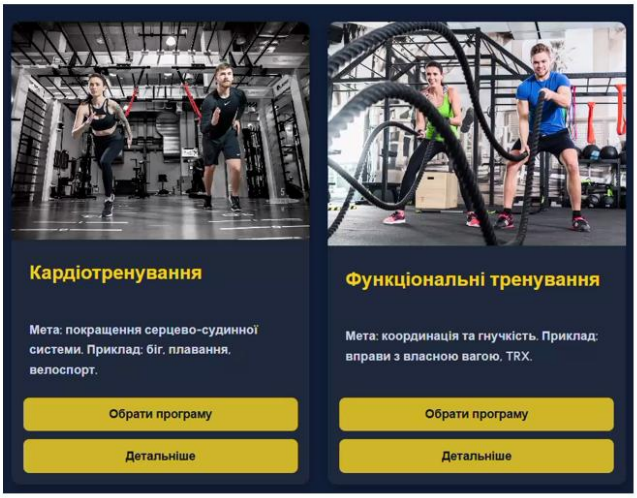


Форма входу в акаунт

ЕКРАННІ ФОРМИ



Панель детальної інформації про програму тренувань



Картки із заготовленими програмами тренувань

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Кулеш І. О., Залива В.В. РОЗРОБКА WEB-ЗАСТОСУНКА “РОЗУМНИЙ ПЛАНУВАЛЬНИК ТРЕНУВАНЬ”. // Матеріали Науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24.04.2025. ДУІКТ, Київ, Україна, С. 207-210

2. Кулеш І. О., Залива В.В. ВИЗНАЧЕННЯ ПЕРЕВАГ ТА НЕДОЛІКІВ ІСНУЮЧИХ WEB-ЗАСТОСУНКІВ. // Матеріали Міжнародної науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». 24.04.2025. ДУІКТ, Київ, Україна, С. (подано до друку)

ВИСНОВКИ

1. Проаналізовано сучасні фітнес-застосунки та їх функціональні можливості.
2. На основі аналізу було визначено вимоги до web-застосунку "Розумний планувальник тренувань“.
3. Підібрано технічно-апаратні технології та програмні засоби для робробки web-застосунку (MongoDB, Express, React, Noode.js, EmailJS API, OpenAI API, GitHub, Bootstrap)
4. Спроєктовано архітектуру повноцінного MERN-застосунку за допомогою генерації власної та вибору існуючих програм тренувань, коригуванням та надсиланням нагадувань)
5. Реалізовано основний функціонал web-застосунку.
6. Налаштовано EmailJS API для автоматичних нагадувань
7. Проведено комплексне тестування web-застосунка на відповідність до визначених вимог.
8. Застосунок було розгорнуто у тестовому режимі.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
<!DOCTYPE html>
<html lang="en">
<head>
   <!-- style-links -->
   <link rel="stylesheet" href="styles/header-
footer.css">
   <link rel="stylesheet" href="styles/main.css">
   <link rel="stylesheet" href="/styles/modal-
auth.css">

   <!-- meta-tags -->
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-
width, initial-scale=1.0">

   <!-- main-font -->
   <link rel="preconnect"
href="https://fonts.googleapis.com">
   <link rel="preconnect"
href="https://fonts.gstatic.com" crossorigin>
   <link
href="https://fonts.googleapis.com/css2?family=Urban
ist:ital,wght@0,100..900;1,100..900&display=swap"
rel="stylesheet">

   <link rel="icon" href="images/icon.png"
type="image/x-icon">

   <title>HealthSync</title>
</head>
<body>
  <header>
   <nav class="navbar">
    <div class="logo">
     <img src="images/logo2.png" alt="HealthSync
Logo">
    </div>

    <ul class="nav-links">
     <li><a href="main.html">Головна</a></li>
     <li><a href="plan/plan.html">Мій план</a></li>
     <li><a
href="programs/programs.html">Програми</a></li>
     <li><a
href="reminder/reminder.html">Нагадування</a></li
>
     <li><a
href="profile/profile.html">Профіль</a></li>
    </ul>

    <div class="nav-buttons">
     <button class="btn login"
id="openLoginModalBtn">Увійти</button>
     <button class="btn signup"
id="openRegisterModalBtn">Реєстрація</button>
    </div>
   </nav>

   <!-- === Модальне вікно реєстрації === -->
   <div id="registerModal" class="modal">
    <div class="modal-content">
     <span class="close"
onclick="document.getElementById('registerModal').st
yle.display='none'">&times;</span>
     <h2>Реєстрація</h2>
     <input type="text" id="registerUsername"
placeholder="Логін" required>
     <input type="email" id="registerEmail"
placeholder="Email" required>
     <div class="password-wrapper">
      <input type="password" id="registerPassword"
placeholder="Пароль" required>
      <button id="toggleRegisterPasswordBtn"
title="Показати пароль">👁</button>
      <button id="generatePasswordBtn"
title="Згенерувати пароль">🔁</button>
     </div>
     <span id="registerPasswordWarning"
class="warning"></span>
     <button
id="submitRegisterBtn">Зареєструватись</button>
    </div>
   </div>

   <!-- === Модальне вікно "Реєстрація успішна"
=== -->
   <div id="successRegisterModal" class="modal">
    <div class="modal-content">
     <h2>Реєстрація успішна</h2>
     <button
onclick="document.getElementById('successRegister
Modal').style.display='none'">Далі</button>
    </div>
   </div>

   <!-- === Модальне вікно входу === -->
   <div id="loginModal" class="modal">
    <div class="modal-content">
     <span class="close"
onclick="document.getElementById('loginModal').styl
e.display='none'">&times;</span>
     <h2>Вхід</h2>
     <input type="text" id="loginUsername"
placeholder="Логін" required>
     <div class="password-wrapper">
      <input type="password" id="loginPassword"
placeholder="Пароль" required>
      <button id="toggleLoginPasswordBtn"
title="Показати пароль">👁</button>
```

```
      </div>
      <button id="submitLoginBtn">Увійти</button>
      <div class="login-extra">
        <button id="forgotPasswordBtn" class="link-
button forgot-password" >Забули пароль?</button>
        <button id="quickLoginBtn" class="quick-
login">Швидкий вхід</button>
      </div>
    </div>
  </div>

  <!-- === Модальне вікно "Вхід успішний" === --
>
  <div id="successLoginModal" class="modal">
    <div class="modal-content">
      <h2>Вхід виконано</h2>
      <button
onclick="document.getElementById('successLoginMo
dal').style.display='none'">Далі</button>
    </div>
  </div>

  <!-- === Модальне вікно "Скидання паролю" ===
-->
  <div id="resetPasswordModal" class="modal">
    <div class="modal-content">
      <span class="close"
onclick="document.getElementById('resetPasswordM
odal').style.display='none'">&times;</span>
      <h2>Скидання паролю</h2>
      <input type="email" id="resetEmail"
placeholder="Ваша пошта" required>
      <button id="resetPasswordSubmitBtn">Скинути
пароль</button>
    </div>
  </div>
 </header>


  <main>
    <!-- Hero Section -->
    <section class="hero">
      <div class="hero-content">
        <h1>Розумні тренування — реальні
результати</h1>
        <p>Отримай індивідуальний фітнес-план,
відстежуй свій прогрес і тримай форму з
HealthSync.</p>
        <button class="cta-button"><a
href="/programs/programs.html">Почати
зараз</a></button>
      </div>

      <div class="hero-image">
        <img src="images/hero-photo.jpg" alt="Fitness
App Preview">
      </div>
    </section>

    <!-- Features -->
    <section class="features">
      <div class="feature">
        <h3>📈 Аналітика прогресу</h3>
        <p>Бачиш свій результат кожного дня</p>
      </div>

      <div class="feature">
        <h3>🤖 Розумні рекомендації</h3>
        <p>AI підлаштовується під тебе</p>
      </div>
      <div class="feature">
        <h3>🔔 Нагадування</h3>
        <p>Більше жодного пропущеного
тренування</p>
      </div>
      <div class="feature">
        <h3>💪 Персоналізація</h3>
        <p>Твій план, твій стиль</p>
      </div>
    </section>

    <!-- How it works -->
    <section class="how-it-works">
      <h2>Як це працює</h2>
      <div class="steps">
        <div class="step">1. Заповнюєш анкету</div>
        <div class="step">2. Отримуєш
персональний план</div>
        <div class="step">3. Тренуєшся, слідкуєш за
результатами</div>
      </div>
    </section>

    <section class="cta-section">
      <h2>Готові покращити своє здоров’я?</h2>
      <p>Зареєструйтесь зараз і почніть свій шлях
до кращого самопочуття з HealthSync.</p>
      <button id="openRegisterModalBtn" class="cta-
button">Зареєструватись</button>
    </section>

    <!-- Modal -->
    <div id="registrationModal" class="modal
hidden">
      <div class="modal-content">
        <span class="close-btn"
id="closeModalBtn">&times;</span>
        <h2>Реєстрація</h2>
        <form class="registration-form">
          <input type="text" placeholder="Ім’я"
required />
          <input type="email" placeholder="Email"
required />
          <input type="password"
placeholder="Пароль" required />
          <button
type="submit">Зареєструватись</button>
        </form>
      </div>
    </div>
  </main>

  <footer>
    <div class="footer-container">
```

```
      <div class="footer-logo">
        <img src="images/logo2.png" alt="HealthSync
Logo">
        <p>Здоров’я та гармонія — завжди під
контролем</p>
      </div>

      <div class="footer-links">
        <h4>Навігація</h4>
        <ul>
          <li><a href="main.html">Головна</a></li>
          <li><a href="plan/plan.html">Мій
план</a></li>
          <li><a
href="programs/programs.html">Програми</a></li>
          <li><a
href="reminder/reminder.html">Нагадування</a></li
>
          <li><a
href="profile/profile.html">Профіль</a></li>
        </ul>
      </div>

      <div class="footer-contact">
        <h4>Контакти</h4>
        <p>Email: support@healthsync.com</p>
        <p>Телефон: +380 99 123 4567</p>
      </div>
    </div>

    <div class="footer-bottom">
      <p>&copy; 2025 HealthSync. Всі права
захищено.</p>
    </div>
  </footer>


 <script src="app.js"></script>
 <script src="java-script/registration.js"></script>
 <script src="java-script/auth.js"></script>

</body>
</html>

document.addEventListener('DOMContentLoaded', ()
=> {
  const user =
JSON.parse(localStorage.getItem('user'));
  const isLoggedIn = localStorage.getItem('loggedIn')
=== 'true';
  const navButtons = document.querySelector('.nav-
buttons');

  // === Функція: оновити header після входу ===
  function showLoggedInHeader(username) {
    navButtons.innerHTML = '';

    const userArea = document.createElement('div');
    userArea.className = 'user-area';

    const profileIcon =
document.createElement('span');
    profileIcon.className = 'profile-icon';
    profileIcon.innerHTML = '<img
src="/images/profile-icon.png" alt="Профіль"
style="width: 30px; height: 30px;">';

    const usernameSpan =
document.createElement('span');
    usernameSpan.className = 'profile-icon' ;
    usernameSpan.textContent = username;

    const logoutBtn =
document.createElement('button');
    logoutBtn.id = 'logoutBtn';
    logoutBtn.textContent = 'Вийти';

    profileIcon.addEventListener('click', () => {
      window.location.href = '/profile/profile.html';
    });
    usernameSpan.addEventListener('click', () => {
      window.location.href = '/profile/profile.html';
    });

    logoutBtn.addEventListener('click', () => {
      localStorage.removeItem('loggedIn');

      navButtons.innerHTML = `
        <button class="btn login"
id="openLoginModalBtn">Увійти</button>
        <button class="btn signup"
id="openRegisterModalBtn">Реєстрація</button>
      `;
```