МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем
_____Юрій КОНДРАТЕНКО
«____»_____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

ІНФОРМАЦІЙНА СИСТЕМА ОЦІНКИ ЗДОРОВ'Я ТА ФІЗИЧНИХ НАВАНТАЖЕНЬ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____Юлія БУГАЙОВА
«___»_____ 2025 р.

Керівник канд. техн. наук, доцент _____ Євген СІДЕНКО
«___»_____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу здобувача

_____ Бугайової Юлії Данилівни _____

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інформаційна система оцінки здоров’я та фізичних навантажень».

Керівник роботи: Сіденко Євген Вікторович, доцент кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент.

Затверджена наказом ЧНУ ім. Петра Могили від «18» листопада 2024 р. № 321.

2. Строк представлення кваліфікаційної роботи «___» _____ 2025 р

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: розробка інформаційної системи, що забезпечує реєстрацію фізичної активності, оцінку стану здоров’я та ведення персонального профілю. Початкові дані включають медичні

показники користувача (вік, вага, зріст), типи фізичних вправ, а також аналіз існуючих фітнес-застосунків.

4. Перелік питань, що підлягають розробці: аналіз предметної області та існуючих рішень; розробка структури бази даних для користувачів, тренуваннь, вправ; проєктування архітектури системи; розробка користувацьких сценаріїв взаємодії; реалізація основного функціоналу: реєстрація/вхід, профіль, додавання вправ; побудова UML-діаграм (випадків використання, класів, послідовності, компонентів); візуалізація інтерфейсу користувача в середовищі Figma; тестування основних функцій застосунку; оцінка ефективності та зручності використання системи.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувачка _____ Юлія БУГАЙОВА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «18» грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

кваліфікаційної роботи

Тема: Інформаційна система оцінки здоров'я та фізичних навантажень.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	18.12.2024	20.12.2024	
2	Аналіз предметної області та постановка задачі	21.12.2024	30.01.2025	
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних систем, щодо інформаційних систем оцінки здоров'я та фізичних навантажень	31.01.2025	01.03.2025	
4	Огляд сучасних вебтехнологій інформаційних систем для підтримки здорового способу життя	02.03.2025	01.04.2025	
5	Реалізація функціональних модулів інформаційної системи з аналізом отриманих результатів	02.04.2025	15.05.2025	
6	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	
7	Корегування роботи за результатами попереднього захисту	29.05.2025	29.05.2025	
8	Другий попередній захист КР на засіданні комісії кафедри	30.05.2025	30.05.2025	
9	Доробка та остаточне оформлення КР	31.05.2025	10.05.2025	
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	11.05.2025	17.05.2025	

Керівник роботи _____ Євген СІДЕНКО

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Юлія БУГАЙОВА

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувачки групи 401 ЧНУ ім. Петра Могили

Бугайової Юлії Данилівни

на тему: **“ ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ОЦІНКИ ЗДОРОВ’Я ТА ФІЗИЧНИХ НАВАНТАЖЕНЬ”**

Актуальність роботи зумовлена потребою у персоналізованих цифрових рішеннях для підтримки здорового способу життя. Більшість наявних фітнес-застосунків мають обмежену гнучкість та не враховують індивідуальні параметри користувача.

Об’єктом роботи є процеси оцінки фізичного здоров’я людини.

Предметом роботи є програмні інструменти для розробки системи для оцінки здоров’я та фізичних навантажень.

Метою роботи є підвищення ефективності фізичного здоров’я людини за рахунок розробки відповідної інформаційної системи.

У роботі проаналізовано існуючі сервіси, визначено вимоги, спроєктовано архітектуру та реалізовано інформаційну систему з модулями реєстрації, профілю, вправ та сетів. Застосунок побудовано на базі JavaScript, React, Firebase та сторонніх API.

Кваліфікаційна робота складається з чотирьох розділів. Загальний обсяг роботи – 83 сторінки. Робота включає 64 рисунків, 11 таблиць і 36 використаних джерел.

Ключові слова: інформаційна система, вебзастосунок, фізична активність, здоров’я, персоналізація, React, Firebase, API, харчування, тренування.

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Yuliia Buhaiova

on the topic: **“INFORMATION SYSTEM FOR HEALTH AND PHYSICAL ACTIVITY ASSESSMENT”**

The relevance of this work is driven by the need for personalized digital solutions that support a healthy lifestyle. Most existing fitness applications offer limited flexibility and do not adequately consider individual user parameters.

The object of the work is the process of assessing a person’s physical health.

The subject of the work is the software tools for developing a system for health and physical activity assessment.

The aim of the work is to improve the effectiveness of physical health assessment by developing an appropriate information system.

The work includes an analysis of existing services, definition of system requirements, design of the architecture, and implementation of a prototype web application. The system includes modules for registration, user profile, exercises, workout sets, and personalized recommendations. The solution is built using JavaScript, React, Firebase, and third-party APIs.

The qualification work consists of four sections. The total volume is 83 pages. The work includes 64 figures, 11 tables, and 36 references.

Keywords: information system, web application, physical activity, health, personalization, React, Firebase, API, nutrition, workout.

ЗМІСТ

ВСТУП

У сучасному суспільстві спостерігається зростаючий інтерес до здорового способу життя, що включає як фізичну активність, так і раціональне харчування. Цей інтерес стимулює розвиток цифрових інструментів, здатних допомагати користувачам у досягненні особистих цілей щодо здоров’я та фізичної форми. Проте більшість існуючих рішень у сфері фітнесу та здоров’я не враховують індивідуальних потреб користувача в повному обсязі – як у плані фізичних навантажень, так і в аспекті харчування.

Актуальність теми зумовлена необхідністю створення адаптивних систем, які можуть формувати персоналізовані рекомендації на основі фізичних параметрів, стилю життя та цілей конкретної людини. Такий підхід дозволяє підвищити ефективність взаємодії користувача з системою та сприяє більшій мотивації до дотримання обраного плану.

Метою даної роботи є підвищення ефективності фізичного здоров’я людини шляхом розробки інформаційної системи, яка забезпечує зручний інтерфейс для моніторингу активності, персоналізовані рекомендації щодо харчування, а також зберігання даних у базі для подальшого аналізу та вдосконалення способу життя. У роботі проведено аналіз предметної області, досліджено існуючі сервіси, визначено функціональні вимоги до системи, а також реалізовано веб-застосунок із сучасним інтерфейсом та адаптивною логікою підбору навантажень і раціону.

Результати даної роботи можуть бути корисними як для широкого кола користувачів, так і для спеціалістів у галузі фітнесу та дієтології, оскільки забезпечують ефективний інструмент підтримки здорового способу життя за допомогою цифрових технологій.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПРОБЛЕМИ ОЦІНКИ ЗДОРОВ’Я ТА ФІЗИЧНИХ НАВАНТАЖЕНЬ

1.1 Опис предметної сфери

Сучасний метод збереження здоров'я та фітнесу включає кілька ключових елементів: постійні фізичні вправи, поживна дієта та уважний підхід до способу життя. У сучасному швидкоплинному світі, де існує безліч варіантів фітнесу та дієти, важливо враховувати конкретні потреби кожної людини при розробці плану вправ і харчування. Це особливо актуально для тих, хто має різні фізичні можливості, інтереси та цілі, такі як зниження ваги, збільшення м'язів або підвищення витривалості. З огляду на ці особисті фактори можна створити більш ефективні програми для досягнення бажаних результатів.

Інформаційна система здоров'я та фізичних вправ, яка обговорюється тут, є програмним інструментом, який автоматично оцінює фізичні атрибути користувача, щоб генерувати індивідуальні рекомендації. Він враховує не тільки стандартні показники, такі як вік, вага та рівень активності, але й конкретні цілі користувача, такі як зниження ваги, нарощування м'язів або підвищення загальної витривалості. Цей метод може значно підвищити ефективність тренування та дієти, враховуючи важливі аспекти здоров'я та індивідуальні характеристики організму. Необхідність створення такої системи обумовлена зростаючим попитом на персоналізовані підходи.

Важливість створення такої системи випливає з зростаючої потреби в персоналізованих методах для сприяння здоров'ю та підвищення фізичної підготовки. Зі зростанням інтересу до активного способу життя, все більше людей використовують мобільні програми та онлайн-інструменти для нагляду за своїм здоров'ям. Однак у багатьох поточних рішеннях все ще бракує всебічного

аналізу та індивідуальних пропозицій, необхідних для досягнення оптимальних результатів.

1.2 Огляд та аналіз наявних аналогів і публікацій

В останні роки мобільні застосунки та інформаційні системи для моніторингу здоров'я та фізичних активностей набули значної популярності завдяки зростаючому інтересу до здорового способу життя та фітнесу. Більшість таких платформ спрямовані на допомогу користувачам у відстеженні їхнього харчування, фізичних вправ та здоров’я в цілому. Однак незважаючи на численні функціональні можливості, існуючі системи мають певні обмеження у наданні індивідуальних рекомендацій, що є важливим аспектом для досягнення високих результатів у фізичному розвитку. Пропоную розглянути системи які найбільш відомі та які активно використовуються користувачами по всьому світу.

Одним з найпопулярніших мобільних застосунків для відстеження харчування та фізичної активності є застосунок **MyFitnessPal**. Він дозволяє користувачам відслідковувати здоров’я і фітнесу. Цей застосунок містить онлайн-калькулятор калорій та планувальник дієти, планування прийняття їжі та відстеження ваги. Основна особливість це велика база даних продуктів і можливість сканувати штрих-коди продуктів, щоб швидко додавати їх до журналу харчування [1].

Ось скріншот інтерфейсу MyFitnessPal, де видно, як відстежуються калорії, що споживаються, а також з’являється можливість додавати продукти за допомогою сканера штрих-кодів (див. рис. 1.1).

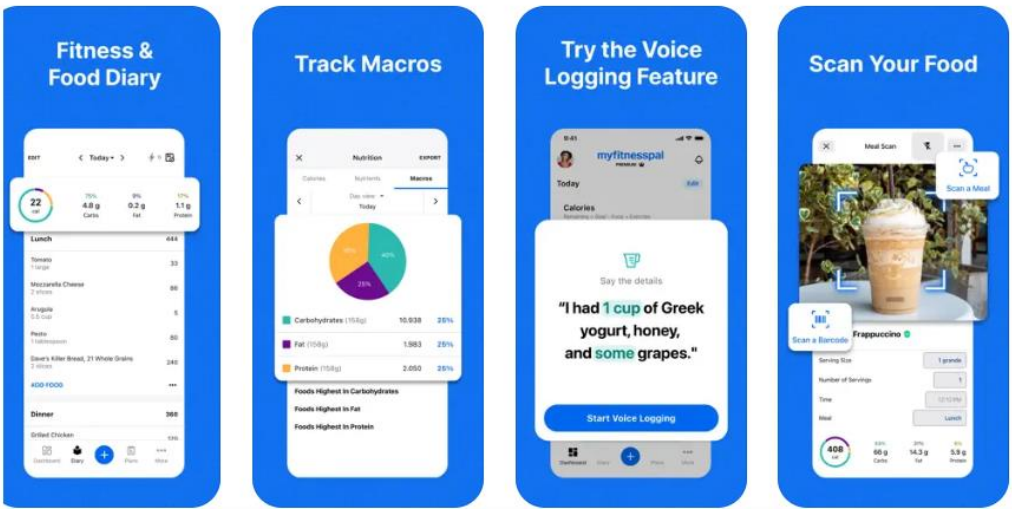

Рисунок 1.1 – Інтерфейс додатку MyFitnessPal [1]

Після проведеного дослідження можна виокремити особливості та недоліки цього застосунку.

Особливості:

– автоматичність відстеження калорій і макроелементів;

– підключення до інших фітнес-пристроїв і додатків;

– відсутність глибокої персоналізації на основі специфічних фізичних параметрів, таких як метаболізм, рівень м’язової маси.

Недоліки:

– додаток не враховує особливості здоров’я користувача, такі як наявність хронічних захворювань або фізичних обмежень;

– в основному всі цікаві та корисні функції платні;

– немає можливості отримати повністю персоналізовані рекомендації на основі індивідуальних фізичних характеристик.

Також одним з популярних брендів які розробляють аксесуари та також пропонують мобільний додаток є **Fitbit**. Основна причина популярності цієї системи є відстеження кроків, сну, серцевого ритму та інших різних параметрів

здоров'я. Як користувач ви можете синхронізувати свої дані між пристроєм та додатком, завдяки чому користувачі в реальному часі можуть отримувати інформацію про фізичний стан. Fitbit також пропонує тренування і планування вправ для людей з різними рівнями підготовки [2].

Використовуючи додаток Fitbit, можна відстежувати прогрес тренувань, зокрема кількість кроків, витрачені калорії, рівень активності за день (див. рис. 1.2).

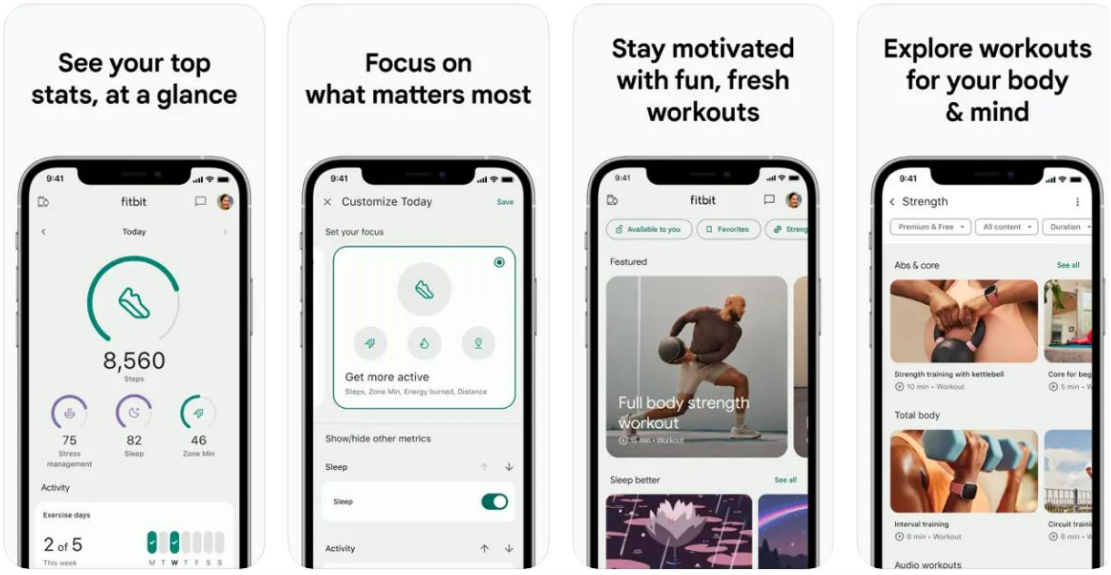

Рисунок 1.2 – Інтерфейс додатку Fitbit [2]

Особливості:

– відстеження активності в реальному часі;

– легкий та зручний інтерфейс, який дозволяє переглядати прогрес по різних параметрах;

– підключення до сторонніх додатків для інтеграції даних, наприклад MyFitnessPal.

Недоліки:

– обмежене персоналізоване налаштування тренувальних програм;

– спеціалізованість на активних користувачів без конкретних фізичних обмежень чи цілей.

А для тих користувачів хто цікавиться спортивними активностями, а саме бігом, велоспортом, трейкінгом або іншими видами фізичної активності є додаток **Strava**, який спеціалізується на моніторингу цих чи інших спортивних активностей. Цей додаток пропонує глибоке інтегрування з іншими пристроями, наприклад GPS-навігатори та фітнес-браслети. Strava надає користувачам можливість конкурувати з іншими учасниками, тим самим це дає змогу мотивувати до змагання та покращення результатів [3].

Далі на скріншоті можна побачити статистику пробіжок або велотренувань з деталями маршруту, швидкості та витрачених калорій (див. рис. 1.3).

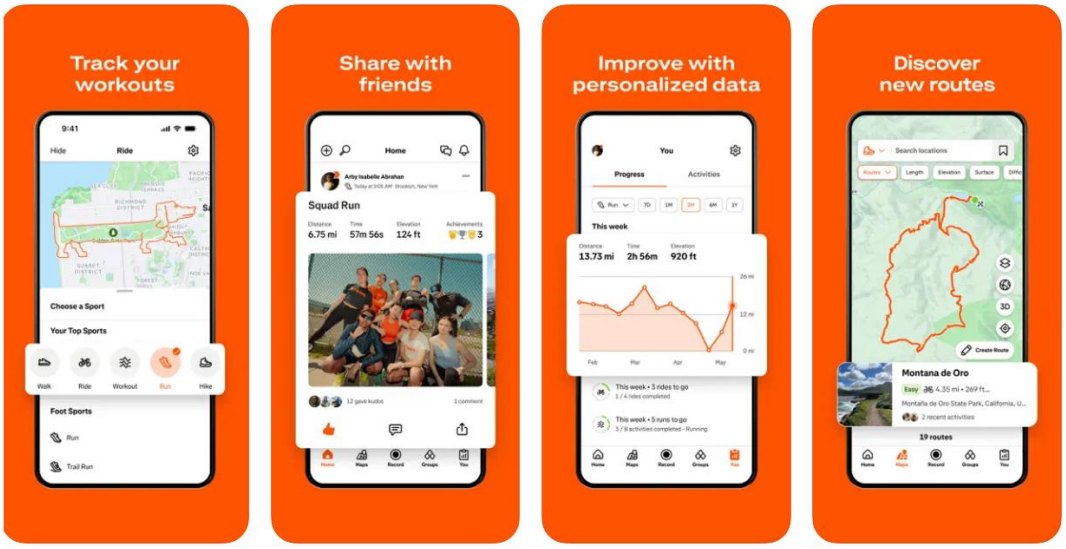

Рисунок 1.3 – Інтерфейс додатку Strava [3]

Основними особливостями цього додатку є:

– система порівняння результатів з іншими користувачами;

– високоточний аналіз фізичної активності;

– підключення до сторонніх додатків та пристроїв.

Недоліки:

– не надаються стратегії схуднення чи набору маси, через обмежену орієнтованість додатку на новачках;

– обмежена персоналізація в плані харчування або розкладу тренувань.

Lose It! є ще одним популярним додатком для моніторингу харчування та контролю калорій. Цей додаток дозволяє користувачам вести журнал харчування, зчитувати калорії і порівнювати їх із встановленими цілями. Lose It! також пропонує персоналізовані рекомендації з харчування, базуючись на цілі користувача, наприклад, схуднення або підтримка здорової ваги [4].

У додатку Lose It! можна побачити графік спожитих калорій і відслідковувати щоденний прогрес (див. рис. 1.4).

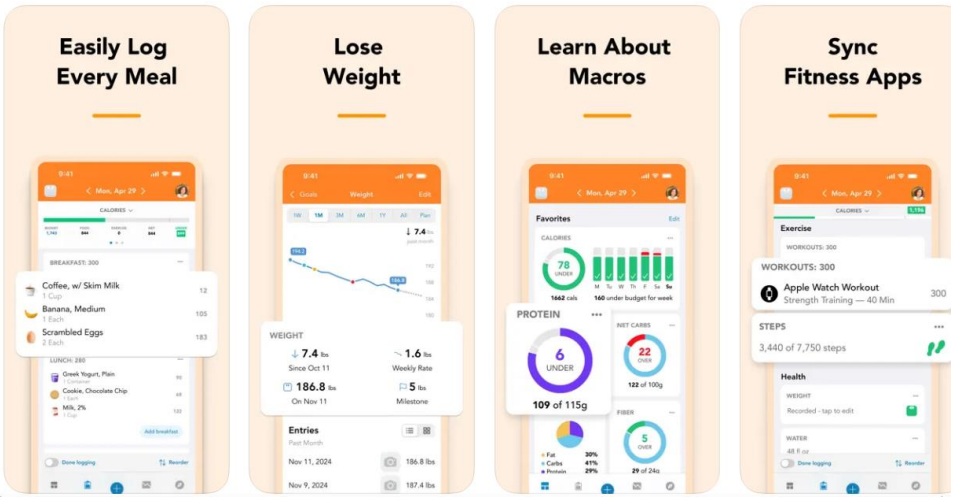

Рисунок 1.4 – Інтерфейс додатку Lose It! [4]

Основними особливостями цього додатку є:

– легкість при відстеженні калорій та макроелементів;

– розгалужена база продуктів;
– можливість створення індивідуального плану харчування.

Недоліки:

– не враховується фізична активність;
– обмежена кількість персоналізованих тренувальних програм.

Для користувачів які полюбляють фітнес такий додаток як **JEFIT**. Цей додаток дозволяє користувачам створювати індивідуальні тренувальні плани, відслідковувати процес і отримувати поради щодо техніки виконання вправ. Ця програма також включає можливість підключення до інших фітнес-пристроїв, тим самим дозволяє отримувати точніші дані про фізичну активність [5].

На цьому скріншоті (див. рис. 1.5) можна побачити що з JEFIT користувач може створювати персоналізовані тренування і відслідковувати виконання кожного з них.

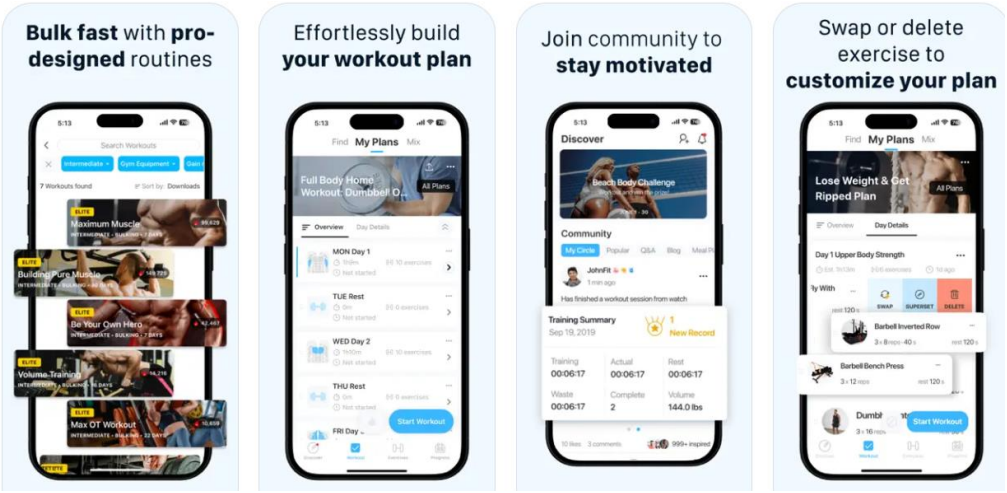

Рисунок 1.5 – Інтерфейс додатку JEFIT [5]

Особливості:

– створення персональних тренувальних планів;

– великий вибір вправ і технік;

– інтеграція з іншими пристроями для точного відстеження прогресу.

Недоліки:

– не надає рекомендацій щодо харчування;

– обмеженні можливості для новачків у фітнесі, бо програма більш орієнтована на досвідчених користувачів.

Для користувачів які мають ціль на схуднення або ж навпаки, набору м'язової маси було розроблено мобільний додаток **YAZIO**. Цей додаток створений для підрахунку калорій і планування харчування. Користувач може обрати власну ціль, а додаток створює індивідуальний план харчування з розрахунком білків, жирів і вуглеводів [6].

З скріншоту можна побачити що на головному екрані відображається щоденний баланс калорій, а також доступ до меню з рецептами (див. рис. 1.6).

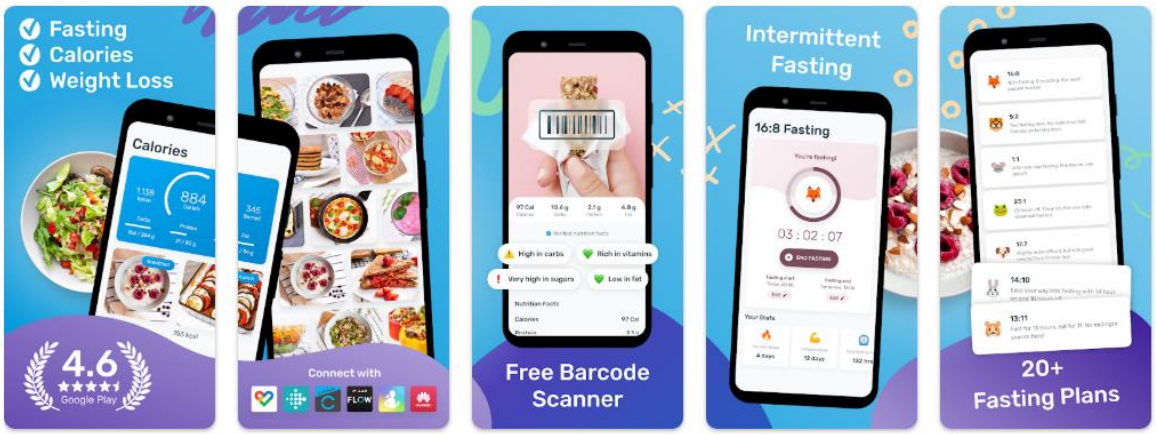

Рисунок 1.6 – Інтерфейс додатку YAZIO [6]

Особливості додатку:

– індивідуальні плахи харчувань з рецептами;

– автоматичне відстеження калорій і макроелементів;

– вбудований планувальник інтервального голодування (посту).

Недоліки:

– обмежена кількість функцій у безкоштовній версії;

– немає комплексної системи відстеження фізичних вправ.

Платформа, яка поєднує фітнес та харчування – **8fit**. Вона пропонує тренування в домашніх умовах, а також персоналізовані плани харчування на повсякдення. Простота використання, мінімальний час тренувань та ефективність вправ робить цю платформу зручною та дієвою [7].

Додаток пропонує 10–20-хвилинні тренування та рецепти з підрахунком калорій (див. рис. 1.7).

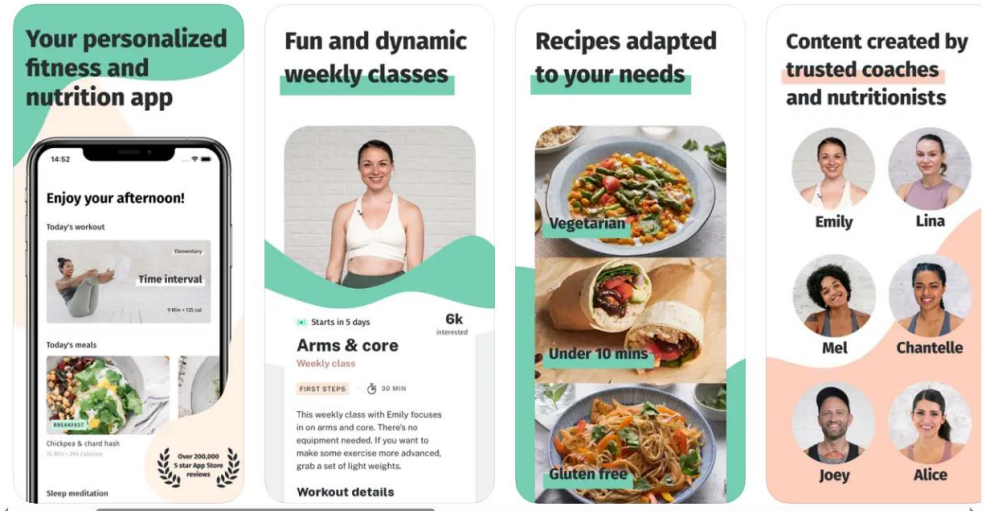

Рисунок 1.7 – Інтерфейс платформи 8fit [7]

Особливості додатку:

– домашні тренування з відеоінструкціями;

– дієтичні плани з урахуванням алергій та переваг користувача;

– оповіщення та моніторинг цілей.

Недоліки:

– обмежена безкоштовна версія;

– менше варіантів налаштувань у порівнянні з професійними системами.

Шведський додаток, який поєднує в собі щоденник харчування, плани дієт і рекомендації щодо здорового способу життя. Який є одним з найбільш привабливих інтерфейсів серед подібних програм – **Lifesum** [8].

На даному скріншоті (див. рис. 1.8) можна побачити, що користувач щодня отримує пропозиції здорової їжі на основі власного харчування.

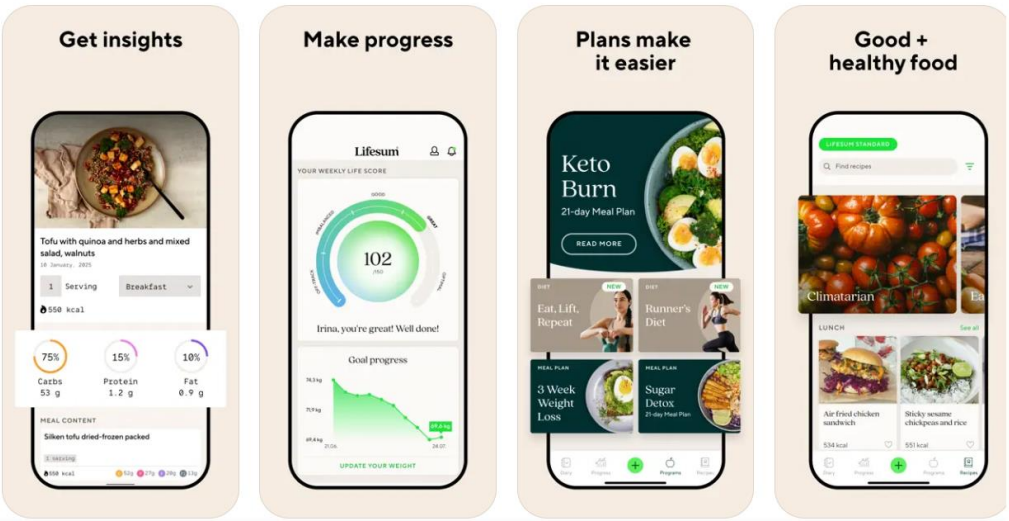

Рисунок 1.8 – Інтерфейс додатку Lifesum [8]

Особливості додатку:

– різноманіття дієт, таких як кето, середземноморська, Палео тощо ;

– вбудований "health test", що оцінює харчові звички;

– індивідуальні поради на щодень.

Недоліки:

– частина функцій доступна лише у преміум-версії;

– обмежене відстеження фізичної активності.

Безкоштовний інструмент від Nike для організації тренувального процесу

– додаток **Nike Training Club**. Містить сотні вправ на силу, гнучкість, витривалість. Відеоуроки з професійними тренерами, включаючи зірок спорту [9].

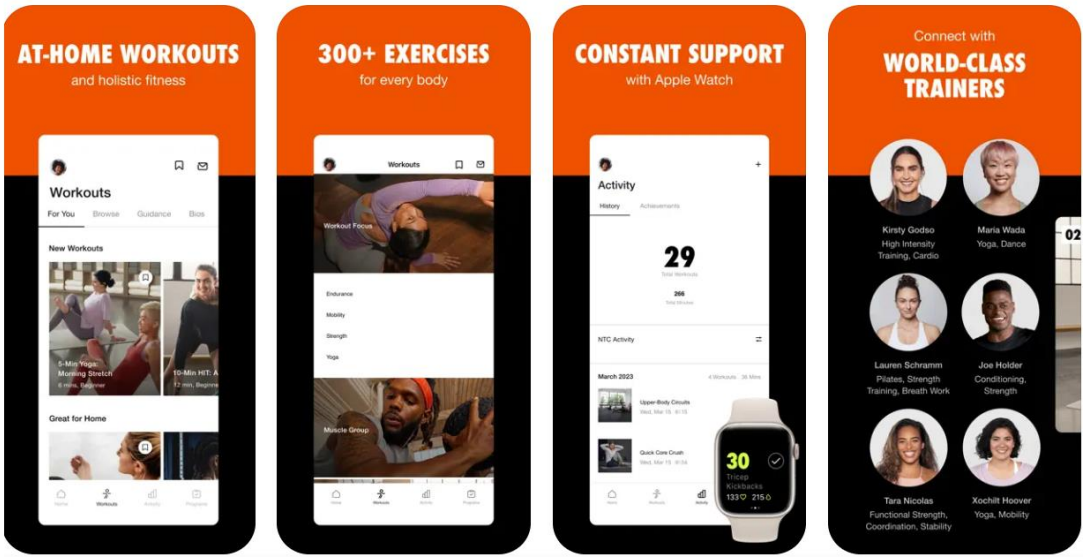

Рисунок 1.9 – Інтерфейс додатку Nike Training Club [9]

Особливості додатку:

– тренування для людей з різним рівнем підготовки;
– фільтрування вправ за тривалістю, складністю та типом вправ;
– синхронізація з Apple Health, Google Fit.

Недоліки:

– відсутні інструменти для відстеження харчування;
– підходить лише для тренувального компоненту.

Аналіз сучасних фітнес-додатків (див. табл. 1.1) показує, що більшість із них орієнтовані на масового користувача та не забезпечують достатню персоналізацію, враховуючи індивідуальні фізичні, медичні чи психологічні особливості. Додатки, такі як MyFitnessPal, YAZIO та Lifesum, охоплюють основні аспекти здорового способу життя, однак їх функції залишаються

загальними. У цьому дослідженні розроблений веб-застосунок дозволяє користувачам створювати індивідуальні тренування, об'єднувати вправи в суперсети, додавати зображення тренажерів і формувати заняття за типами (силові, кардіо, розтяжка), що забезпечує більшу персоналізацію. Новітні AI-платформи, як зазначає Американський коледж спортивної медицини (ACSM), використовують великі дані для створення фітнес-програм, адаптованих до індивідуальних потреб користувачів [10]. Крім того, гаджети для збору чутливих даних, таких як рівень активності та частота серцевих скорочень, дозволяють ще більше персоналізувати підхід до здоров'я [11].

Таблиця 1.1 – Таблиця порівняння функцій аналогів фітнес-додатків

Функція	**My FitnessPal**	**YAZIO**	**8fit**	**Lifesum**	**Nike Training Club**	**Fitbit**	**JEFIT**	**Lose It!**	**Strava**
Підрахунок калорій	✓	✓	✓	✓	✗		✗	✓	✗
Трекінг харчування	✓	✓	✗	✓	✗		✗	✓	✗
План харчування	✓	✓	✓	✓	✗		✗	✓	✗
Програми тренувань	✗	✓	✓	✓	✓		✓	✗	✓
Персоналізація плану	✓	✓	✓	✓	✗		✓	✓	✓
Відстеження прогресу	✓	✓	✓	✓	✓		✓	✓	✓
Синхронізація з пристроями	✓	✓	✗	✓	✓		✓	✓	✓
Соціальні функції	✓	✗	✗	✗	✓		✓	✓	✓
Відстеження активності	✓	✓	✓	✓	✓		✓	✓	✓

Безкоштовна версія	✓	✓	✓	✓	✓	✓	✓	✓	✓
Платна підписка	✓	✓	✓	✓	✓	✓	✓	✓	✓

1.3 Постановка задачі

Актуальність теми

У сучасному світі дедалі більше людей приділяють увагу своєму здоров'ю та фізичній активності. Високий ритм життя, стрес, недостатня кількість вільного часу стали факторами, що часто змушують відкладати турботу про власне тіло на другий план. Водночас цифрові технології дедалі активніше інтегруються у повсякденне життя та відкривають нові можливості для самоконтролю.

Фітнес-додатки та онлайн-сервіси, що дозволяють стежити за фізичною активністю, сном, харчуванням та іншими показниками здоров'я, стали звичним інструментом для користувачів. Проте більшість таких систем мають обмежену функціональність і не завжди враховують індивідуальні параметри кожного користувача. Часто вони зводяться до базового моніторингу без глибокої адаптації під особисті потреби.

У зв'язку з цим, актуальною є розробка систем, що дозволяють гнучко відображати, зберігати та аналізувати інформацію про фізичний стан людини. Саме інструменти, орієнтовані на індивідуальний підхід до оцінки фізичного здоров'я, мають потенціал підвищити ефективність самоспостереження та контроль за станом організму. Тема створення таких інформаційних систем є перспективною та потребує глибокого технічного й практичного опрацювання.

Метою цієї роботи є підвищення ефективності фізичного здоров'я людини шляхом розробки інформаційної системи, яка забезпечує зручний інтерфейс для

моніторингу активності, персоналізовані рекомендації щодо харчування, а також зберігання даних у базі для подальшого аналізу та вдосконалення способу життя.

Об’єктом цієї роботи є процеси оцінки фізичного здоров’я людини.

Предметом роботи є програмні інструменти та технології для розробки системи, що дозволяє здійснювати оцінку здоров’я та фізичних навантажень.

Робота спрямована на розробку системи для контролю фізичної активності, моніторингу стану організму, формування індивідуальних планів тренувань та ведення здорового способу життя за допомогою цифрових технологій. Значну увагу приділено огляду існуючих засобів веброзробки та технологіям зберігання і обробки даних.

Щоб досягти поставленої мети, потрібно виконати кілька **завдань**:

– проаналізувати існуючі мобільні застосунки, що стосуються здоров’я, харчування, фізичної активності, відстеження прогресу тощо;

– з’ясувати, які функції користувачі вважають найбільш корисними, а які – зайвими чи незручними;

– визначити ключові проблеми, з якими стикаються люди при користуванні такими застосунками;

– сформулювати вимоги до нового застосунку, який буде зручним, зрозумілим і максимально персоналізованим;

– розробити концепцію або прототип такого застосунку з урахуванням зібраної інформації.

Висновки до розділу 1

У першому розділі розглянуто передумови створення інформаційної системи для оцінки фізичного здоров’я та навантажень. Обґрунтовано актуальність теми в умовах зростання інтересу до здорового способу життя й

необхідності персоналізованих цифрових рішень.

Сформульовано об’єкт дослідження – процеси оцінки фізичного стану людини, а також предмет – програмні засоби для створення відповідної системи. Визначено мету роботи, що полягає у підвищенні ефективності підтримки фізичного здоров’я за допомогою сучасної інформаційної системи.

Проведено огляд існуючих мобільних та вебсервісів у сфері здоров’я та фітнесу, виявлено основні недоліки в персоналізації, адаптації під користувача та взаємодії з даними. Визначено потребу у створенні більш гнучкого інструменту, який дозволятиме враховувати індивідуальні цілі, особливості та вподобання користувача.

Отримані результати закладають підґрунтя для наступних етапів проєктування й реалізації інформаційної системи, орієнтованої на покращення якості життя користувача через підтримку фізичної активності та моніторинг здоров’я.

2 МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ОЦІНКИ ЗДОРОВ’Я ТА ФІЗИЧНИХ НАВАНТАЖЕНЬ

2.1 Методи для вирішення поставленої задачі

Розробка інформаційної системи для оцінки здоров’я та фізичних навантажень вимагає використання комплексу методів, що дозволяють забезпечити індивідуальний підхід до користувача, а також забезпечити зручний інтерфейс, ефективну взаємодію з даними та гнучку логіку формування рекомендацій. Задача не зводиться лише до відображення списків вправ або рецептів – система повинна аналізувати потреби користувача, забезпечувати пошук, відбір та групування інформації, а також її виведення у формі, зручній для сприйняття.

У межах цього проєкту були застосовані наступні методи:

а) метод аналізу предметної області: на початковому етапі було проведено глибоке вивчення сучасних тенденцій у сфері фітнес-технологій, електронного здоров’я та харчового консалтингу. Цей аналіз дозволив визначити основні вимоги до майбутньої системи, а також виявити типові потреби користувачів. Була побудована узагальнена структурна модель інформаційної системи, яка включає основні функціональні блоки: модуль роботи з вправами, модуль харчування, модуль пошуку, модуль рекомендацій та інтерфейс користувача;

б) метод фільтрації даних: оскільки користувачам необхідно швидко знаходити інформацію про фізичні вправи або рецепти, доцільно використовувати метод фільтрації – тобто відбір даних за заданими критеріями. Цей метод реалізований у вигляді функціоналу пошуку за ключовими словами (наприклад, назва вправи, м’язова група, рівень складності, назва рецепту,

інгредієнти тощо), а також відбору за категоріями, такими як «вправи для рук», «низькокалорійні рецепти», «швидке приготування» тощо;

в) метод категоризації та угрупування: щоб надати користувачам можливість швидко орієнтуватися у великому обсязі даних, було використано метод категоризації. Вправи та рецепти структуровані у відповідні категорії – за частинами тіла (руки, ноги, спина), за рівнем складності (новачок, середній рівень, просунутий), за тривалістю виконання, тощо. Цей підхід дозволяє покращити користувацький досвід та зменшити час на пошук необхідної інформації;

г) метод візуалізації інформації: для зручності сприйняття результатів користувачами застосовано метод візуалізації, що передбачає подання інформації у вигляді карток, слайдерів та інтерактивних блоків. Завдяки використанню компонентів на основі бібліотеки Material UI та кастомного SCSS, користувачі мають можливість переглядати ключову інформацію – зображення, назву вправи, час приготування рецепту – ще до переходу на повну сторінку;

д) метод API-інтеграції: щоб забезпечити надання актуальної та перевіреної інформації, було застосовано метод інтеграції із зовнішніми джерелами даних (API). Зокрема, реалізовано роботу з двома основними публічними API:

1) ExerciseDB API – для отримання інформації про фізичні вправи, м’язові групи та необхідне обладнання;

2) Spoonacular API – для отримання рецептів, харчових характеристик та описів приготування страв.

API-запити реалізовані у форматі REST через асинхронні функції на JavaScript із використанням fetch() та відповідної конфігурації заголовків для авторизації;

е) метод контролю коректності даних: з метою запобігання помилкам при завантаженні даних або неправильному форматуванні відповідей API застосовується перевірка типів та перевірка на наявність даних. Наприклад, перед виведенням блоку вправ перевіряється, чи масив вправ не є порожнім, чи існує поле name, image, target тощо. Це дозволяє уникнути критичних помилок та забезпечити стабільну роботу інтерфейсу;

ж) метод компонування інтерфейсу: в основі побудови інтерфейсу лежить компонентний підхід, який дозволяє розділити функціональність на незалежні логічні блоки (компоненти): SearchExercises, ExerciseCard, RecipeShortInfo, Scroll, QuickRecipes тощо. Це спрощує підтримку коду, дозволяє повторно використовувати елементи на різних сторінках та забезпечує швидку адаптацію до змін.

У процесі вирішення задачі створення інформаційної системи для оцінки фізичного стану та формування рекомендацій було застосовано комплекс методів, що охоплюють як аналіз предметної області, так і технічні методи фільтрації, категоризації, інтеграції з API та побудови інтерфейсу. Це дозволило сформувати стійку архітектуру, що забезпечує ефективну взаємодію з користувачем і відповідає вимогам зручності, швидкодії та гнучкості.

2.2 Технології розробки системи

Під час розробки інформаційної системи для оцінки здоров’я та фізичних навантажень важливу роль відіграє правильний вибір технологій. Адже від того, наскільки вдало підібрані інструменти, залежить і зручність розробки, і продуктивність системи, і якість взаємодії з користувачем. У цьому розділі розглянуто ті програмні засоби, мови, бібліотеки та API, які було використано для створення інформаційної системи, а також причини їх вибору.

Уся система створюється як односторінковий вебзастосунок (SPA – Single Page Application), що дозволяє уникнути постійного перезавантаження сторінок, роблячи користування сайтом плавним, швидким і приємним.

У проєкті були використані такі ключові технології:

– **JavaScript**: мова програмування, яка є основною для взаємодії між елементами інтерфейсу та логікою додатка. Завдяки її широким можливостям, стало можливим реалізувати складні механізми пошуку, фільтрації та роботи з API. Синтаксис JavaScript дозволяє швидко створювати як прості функції, так і складну взаємодію між компонентами, як зазначається в офіційній документації [12];

– **React**: сучасна JavaScript – бібліотека, яка дозволяє будувати інтерфейс з окремих незалежних компонентів. Це значно спрощує структурування коду, підвищує його гнучкість і повторне використання. Наприклад, один і той самий компонент картки вправи або рецепту може використовуватись у кількох частинах сайту. React також підтримує віртуальний DOM, що забезпечує високу швидкість оновлення елементів на сторінці, згідно з офіційною документацією [13];

– **HTML5 та CSS3**: класичні мови розмітки і стилів, які забезпечують структуру сторінок і їх зовнішній вигляд. HTML використовується для формування базової структури елементів, а CSS – для оформлення, адаптації під різні розміри екранів та візуальної привабливості. Ці технології детально описані у відповідних специфікаціях [14-15];

– **SCSS (Sassy CSS)**: препроцесор, який дозволяє писати CSS з використанням змінних, вкладеності, міксинів та інших розширених можливостей. Це значно покращує організацію стилів у великих проєктах і спрощує подальше редагування, як зазначено в офіційному довіднику [16];

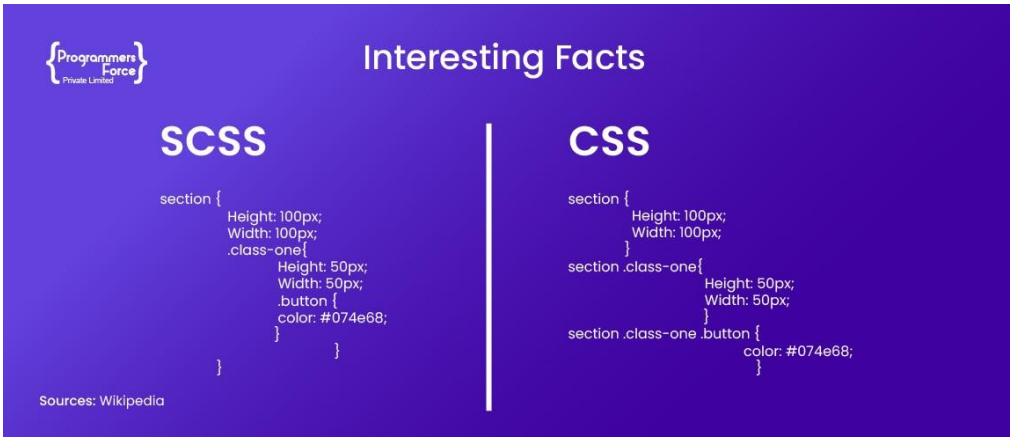

Рисунок 2.1 – Порівняння препроцесору SCSS з мовою розмітки CSS [35]

– **Material UI (MUI)**: набір компонентів інтерфейсу, який відповідає принципам дизайну Google Material Design. Завдяки MUI стало можливо швидко створити сучасний, адаптивний і дружній до користувача інтерфейс, який виглядає гармонійно на різних пристроях. Наприклад, такі елементи як кнопки, текстові поля, меню та слайдери вже мають готову стилізацію і можуть бути налаштовані під потреби проєкту, як зазначається в документації MUI [17];

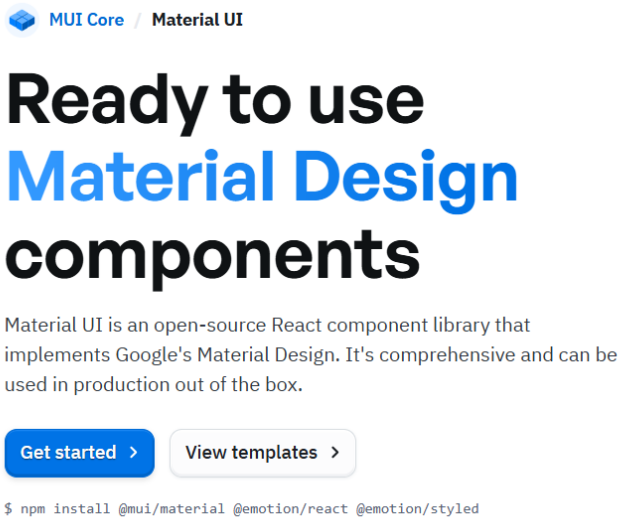

Рисунок 2.2 – Скріншот імплементації Material UI з офіційного сайту MUI [17]

– **React Router**: бібліотека для маршрутизації, яка забезпечує перехід між сторінками без перезавантаження браузера. У нашому проєкті вона відповідає за переміщення між домашньою сторінкою, сторінками вправ, детальною інформацією про конкретну вправу або рецепт тощо, як описано в офіційному гіді [18];

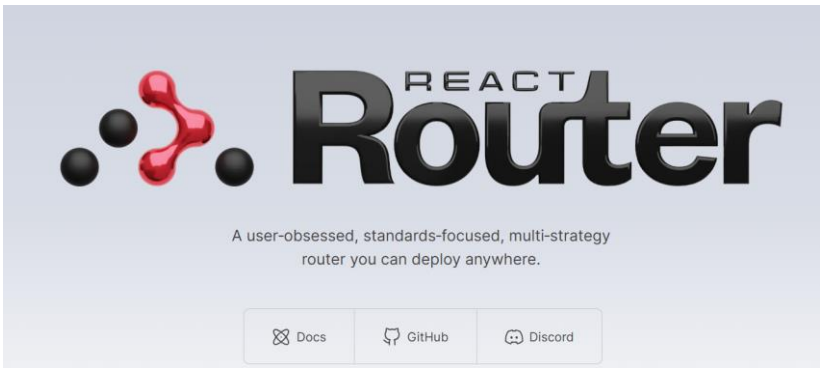

Рисунок 2.3 – Скріншот з офіційної сторінки ReactRouter [18]

– **Flickity**: бібліотека для створення інтерактивних слайдерів та прокручуваних блоків. З її допомогою реалізовано відображення рекомендованих або швидких рецептів у вигляді прокручуваних каруселей, як зазначається на сторінці документації Flickity [19];

Рисунок 2.4 – Скріншот з офіційної сторінки Flickity [19]

– **Зовнішні API (Application Programming Interface)**: використання сторонніх джерел даних дозволило значно розширити можливості додатку без необхідності створювати власні бази даних вправ чи рецептів. У проєкті використовувалися:

1) **ExerciseDB API** – для отримання переліку фізичних вправ, м'язових груп, інформації про цільові м'язи та обладнання [20];

2) **Spoonacular API** – для отримання інформації про рецепти, час приготування, інгредієнти, калорійність, а також для генерації жартів і цікавих фактів про їжу [21];

3) **YouTube Search API** – для виведення відеоуроків з технікою виконання вправ на основі ключових слів [22].

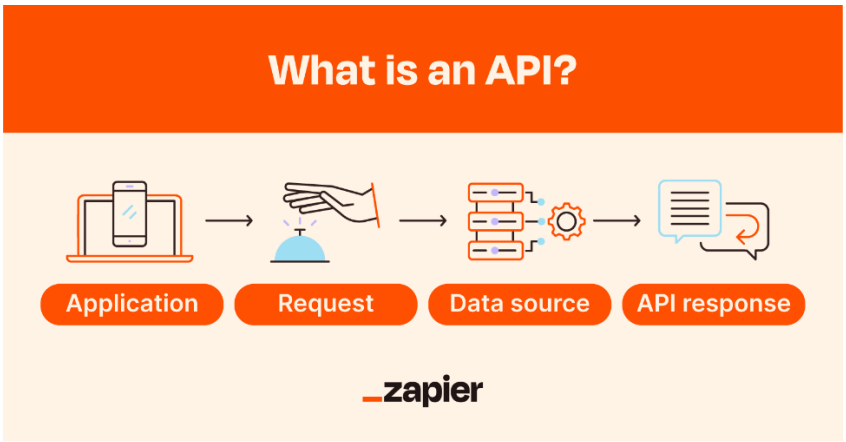

Рисунок 2.5 – Рисунок для розуміння як працює API [36]

– **Fetch API**: метод вбудованої роботи з HTTP-запитами в браузері. Він використовується для взаємодії з API – надсилання запитів та обробки відповідей. Асинхронність у поєднанні з async/await забезпечує зрозумілий та структурований код, як зазначено в MDN [23];

– **Git та GitHub**: система контролю версій, яка дозволила відстежувати зміни, працювати з різними гілками та зберігати резервні копії проєкту. Репозиторій проєкту знаходиться у відкритому доступі на GitHub, що також дозволяє легко публікувати застосунок через GitHub Pages [24-25];

– **GitHub Pages**: сервіс для безкоштовного хостингу односторінкових вебдодатків. У моєму випадку застосунок автоматично компілюється і деплоїться з гілки main, що дозволяє швидко оновлювати публічну версію сайту [26];

– **Node.js та npm (Node Package Manager)**: хоча проєкт є фронтенд-застосунком, підключення пакетів через npm є критично важливим. Зокрема, через npm встановлювались бібліотеки React, MUI, React Router, SCSS, Flickity, Axios, а також утиліти для форматування, перевірки коду та збору проєкту. Інформація наведена згідно з офіційною документацією Node.js [27-28];

– **ReactDOM та JSX**: JSX – це синтаксис, що дозволяє писати HTML у JavaScript. Завдяки йому можливо створювати шаблони елементів з логікою прямо у функціональних компонентах. А ReactDOM дозволяє рендерити ці компоненти на сторінці, як пояснюється у React-документації [29-30].

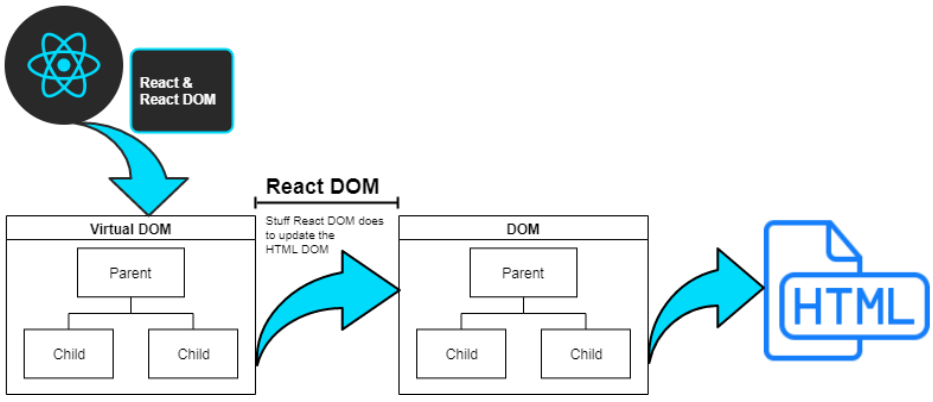

Рисунок 2.6 – Процес роботи ReactDOM та JSX [37]

Розробка та підтримка проєкту здійснюється у середовищі **Visual Studio**

Code – це сучасний, безкоштовний текстовий редактор, що надає широкий набір інструментів для зручної розробки JavaScript/React-застосунків. Серед основних переваг – велика кількість розширень (наприклад, ESLint, Prettier, Live Server, SCSS IntelliSense), підтримка гіт-репозиторіїв прямо з вікна редактора, автоматичне підсвічування синтаксису, автодоповнення коду та інтеграція з терміналом [31].

Під час розробки активно використовувався вбудований термінал VS Code для виконання таких команд:

– запуск локального сервера *(npm start)*;
– встановлення залежностей *(npm install)*;
– форматування коду *(prettier --write .);*
– деплой на GitHub Pages *(npm run deploy)*.

Visual Studio Code також дозволив інтегрувати контроль версій через Git – це значно полегшило збереження прогресу, створення гілок, тестування нових ідей і відновлення коду в разі помилок.

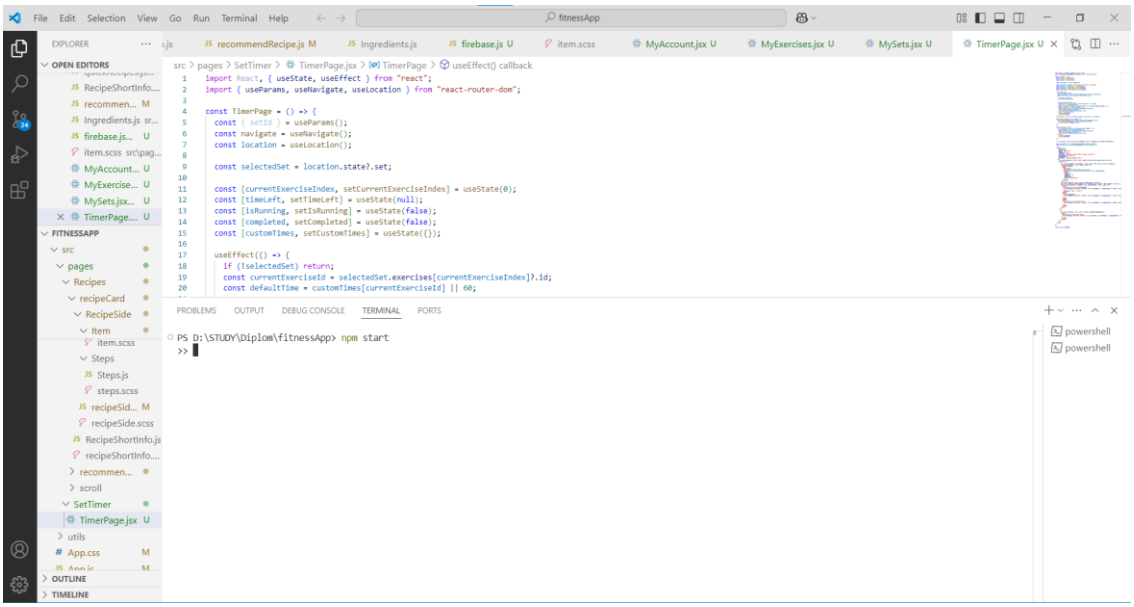

Рисунок 2.7 – Зображення інтерфейсу VS Code

Усі ці технології працюють у тісній зв’язці, утворюючи зручне середовище для створення потужного вебзастосунку, який легко підтримувати та оновлювати. Завдяки сучасним інструментам розробки вдалось досягти високої швидкості відгуку сайту, гарної адаптивності до екранів різних пристроїв і приємного зовнішнього вигляду.

Висновки до розділу 2

У цьому розділі було детально розглянуто методи та технології, які стали основою для реалізації інформаційної системи оцінки здоров’я та фізичних навантажень. Була проведена ретельна характеристика прикладних підходів, таких як аналіз предметної області, фільтрація даних, категоризація та візуалізація результатів, які дозволили створити логіку системи, орієнтовану на потреби користувача.

Також були обґрунтовані вибрані програмні засоби – зокрема, використання зв’язки JavaScript + React дало змогу створити швидкий, гнучкий і масштабований фронтенд, тоді як технології типу SCSS, MUI та React Router забезпечили сучасний вигляд та зручну навігацію між сторінками. Значну роль у реалізації динамічної частини системи відіграли зовнішні API, які дозволили інтегрувати базу вправ, рецептів, рекомендацій та відеоматеріалів без потреби у власній серверній частині.

Особливу увагу було приділено інструментам розробки. Робота здійснювалась у середовищі Visual Studio Code, що надало змогу ефективно керувати проєктом, підключати бібліотеки через npm та виконувати тестування.

Таким чином, обрані методи та інформаційні технології повністю відповідають поставленій меті та забезпечують необхідний функціонал для

подальшого розвитку системи, що зможе надавати користувачам персоналізовані поради щодо здоров’я та фізичних навантажень.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОЦІНКИ ЗДОРОВ'Я ТА ФІЗИЧНИХ НАВАНТАЖЕНЬ. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Опис вхідних даних та структури системи

Розроблена інформаційна система є багатофункціональним веб-застосунком для підтримки здорового способу життя, який дозволяє користувачам отримувати індивідуальні рекомендації щодо фізичних вправ та раціону харчування на основі особистих даних. Основною метою системи є створення персоналізованого досвіду, що базується на фізіологічних параметрах користувача, його цільових показниках, уподобаннях, а також збережених діях (вподобані вправи, створені комплекси тощо).

Інформаційна система включає наступні основні модулі:

– модуль аутентифікації (реєстрація/вхід);

– модуль управління особистим профілем;

– модуль управління вправами та тренуваннями;

– модуль збереження користувацьких даних (вподобання, створені вправи тощо);

– інтерфейс адміністратора (опціонально – для модерації контенту).

Архітектура системи побудована досить просто і зручно: є веб-інтерфейс, з яким працює користувач, і є сервер, який обробляє запити від цього інтерфейсу. Вони «спілкуються» між собою через спеціальні запити – REST API. Щоб зберігати всі дані користувачів, включаючи профілі, улюблені вправи, персональні плани тощо, використовується сучасна база даних без чіткої структури – NoSQL, наприклад Firebase Firestore. Вона добре підходить для таких задач, бо працює швидко, легко масштабується і дозволяє зберігати багато різної персональної інформації без ускладнень.

Інформаційна система складається з кількох основних частин, які разом утворюють зручний сервіс для користувача. Вона працює за принципом клієнт-серверної архітектури, тобто є дві основні складові.

Клієнтська частина (веб-інтерфейс). Це те, що бачить користувач – вебсторінка або застосунок. Через неї користувач може реєструватися, входити в особистий кабінет, заповнювати дані про себе (наприклад, вік, стать, цілі – схуднення, набір маси, підтримка форми), шукати вправи, додавати їх у «улюблене», створювати власні вправи або комбіновані сети (набори вправ), переглядати рекомендовані рецепти тощо.

Інтерфейс зроблений максимально зручним та зрозумілим, щоб навіть новий користувач міг легко зорієнтуватися. Усі дії виконуються без необхідності перезавантаження сторінки – швидко й комфортно.

Серверна частина (бекенд). Це те, що «під капотом» – тобто програма на сервері, яка приймає всі запити від користувача, обробляє їх і повертає результати. Наприклад, коли користувач натискає кнопку «Зберегти вправу в улюблене», запит надсилається на сервер, який зберігає це в базу даних і повертає підтвердження.

Усі запити між клієнтом і сервером відбуваються через так званий REST API – це набір правил, за якими вони взаємодіють. REST дозволяє структурувати роботу системи так, щоб її було легко підтримувати і розвивати в майбутньому.

База даних (зберігання інформації). Уся інформація про користувачів, вправи, рецепти, улюблене, історію виконаних вправ, плани тренувань тощо зберігається у NoSQL-базі даних, наприклад у Firebase Firestore. Це сучасна хмарна система, яка зберігає дані у вигляді документів, а не таблиць. Такий підхід дозволяє легко зберігати і обробляти дуже різні типи інформації – наприклад,

дані одного користувача можуть відрізнятися від іншого, і система все одно працюватиме стабільно.

Це рішення обране тому, що:

– воно швидко працює;

– добре підходить для зберігання персоналізованої інформації;

– легко масштабується, тобто зростання кількості користувачів не стане проблемою.

Вхідні дані системи. Система працює з різними типами вхідних даних, які надходять від користувача або завантажуються з бази. Умовно їх можна поділити на кілька груп.

Дані під час реєстрації та створення профілю. Коли новий користувач вперше заходить у систему, він створює обліковий запис, де зазначає такі дані, які наведені у таблиці 3.1.

Таблиця 3.1 – Дані користувача

Поле	Тип даних	Опис
Ім’я	Текст	Ім’я користувача
Вік	Ціле число	Вік у роках
Стать	Перелік (ч/ж)	Біологічна стать
Зріст	Ціле число (см)	Для розрахунку ІМТ
Вага	Ціле число (кг)	Поточна вага
Рівень підготовки	Вибір із 3-х	Початковий / середній / просунутий
Мета	Перелік	Схуднення / Набір маси / Підтримка форми

Ці дані зберігаються в базі й використовуються для формування персональних рекомендацій.

Дані вправ. Система містить каталог вправ, кожна з яких має структурований опис. Також користувачі можуть створювати власні вправи (див.

табл. 3.2);

Таблиця 3.2 – Дані вправ

Поле	Тип	Опис
Назва	Текст	Назва вправи
Тип	Перелік	Силова / кардіо / гнучкість / розтяжка
Складність	Перелік	Початкова / середня / висока
М’язи	Список	М’язові групи, що задіяні
Інструкція	Текст	Покроковий опис виконання
Зображення/відео	URL	Посилання на медіаконтент

Інтерактивні дії користувача. У процесі користування сервісом система також отримує інші вхідні дані:

– вибрані вправи (які користувач додав у "Улюблене");

– створені користувачем вправи (назва, опис, тривалість, цільові групи м’язів, інструкції, зображення/відео);

– збережені сети (набори вправ) – наприклад, «Ранкове тренування», «Швидке кардіо», «Сила для ніг»;

– оцінки та коментарі до вправ (якщо передбачено);

– історія активності (які вправи виконувались, як часто).

Внутрішні дані системи. Це фонові або технічні дані, які зберігаються й обробляються автоматично:

– ID користувачів, ID вправ, ID наборів;

– дата й час виконання вправ;

– статистика (кількість переглядів, збережень у «улюблене», створених вправ тощо).

Як виглядає профіль користувача.

Особистий профіль – це окрема сторінка, яка відображає індивідуальну інформацію користувача:

– блок з персональними даними: ім'я, вік, стать, рівень підготовки, ціль;

– розділ «Улюблене»: збережені вправи, які користувач може швидко переглянути або запустити;

– розділ «Мої вправи»: тут знаходяться вправи, створені самим користувачем;

– розділ «Мої сети»: створені користувачем набори вправ, які можна зберігати та запускати.

3.2 Проєктування інтерфейсу та структури

На цьому етапі було розроблено концепцію майбутньої системи: визначено, які основні компоненти вона має містити, як вони між собою взаємодіють, які дані будуть зберігатися, і як забезпечити зручність та безпеку для кінцевого користувача. Уся система створюється так, щоб бути простою для користування, гнучкою для розширення і надійною у роботі.

Загальна структура. Інформаційна система побудована за клієнт-серверною архітектурою. Це означає, що існує дві основні частини:

– клієнт – вебінтерфейс, через який користувач взаємодіє з системою: реєструється, переглядає вправи та рецепти, додає улюблене, створює сети тощо;

– сервер – виконує всю логіку, обробляє запити клієнта, зберігає дані в базі, генерує персональні рекомендації, перевіряє права доступу та автентифікацію.

Вся комунікація між клієнтом і сервером відбувається через REST API, що дозволяє зручно обмінюватись даними у форматі JSON.

Структура бази даних. Для зберігання даних використовується NoSQL-база Firebase Firestore, яка дозволяє зберігати інформацію у вигляді колекцій і документів. Такий підхід є гнучким і добре підходить для систем з користувацьким контентом.

Основні колекції:

– users – інформація про користувачів: ім'я, вік, стать, цілі, рівень фізичної підготовки, email, список улюбленого (див. табл. 3.3);

Таблиця 3.3 – Колекція users (користувачі)

Поле	Тип	Опис
userId	string	Унікальний ідентифікатор користувача (UID)
name	string	Ім'я користувача
age	number	Вік
gender	string	Стать (чол./жін./інше)
weight	number	Вага (кг)
height	number	Зріст (см)
goal	string	Ціль (схуднення, маса, підтримка)
level	string	Рівень підготовки
favorites	array	ID улюблених вправ/сетів

– exercises – вправи з детальним описом: назва, тип, складність, інструкція, м'язові групи, автор (див. табл. 3.4);

Таблиця 3.4 – Колекція exercises (вправи)

Поле	Тип	Опис
exerciseId	string	Унікальний ID вправи
name	string	Назва вправи
description	string	Детальний опис
muscleGroup	string	Цільова м'язова група
difficulty	string	Рівень складності (легка, середня тощо)
authorId	string	ID користувача, якщо це його вправа
createdAt	timestamp	Дата створення

– sets – зібрані користувачем сети з вправ: назва, список вправ, тривалість, рівень (див. табл. 3.5);

Таблиця 3.5 – Колекція sets (набори вправ)

Поле	Тип	Опис
setId	string	Унікальний ID набору
name	string	Назва набору
exercises	array	Список ID вправ

Кінець таблиці 3.5

totalTime	number	Загальний час тренування (хв)
difficulty	string	Складність
authorId	string	Хто створив

– recipes – рецепти: інгредієнти, калорійність, призначення (схуднення, набір маси тощо) (див. табл. 3.6);

Таблиця 3.6 – Колекція recipes (рецепти)

Поле	Тип	Опис
recipeId	string	Унікальний ID рецепту
name	string	Назва
ingredients	array	Інгредієнти
calories	number	Калорійність
goalType	string	Призначення (схуднення, маса)
steps	string	Інструкції приготування

– favorites – улюблені вправи/сети користувача (див. табл. 3.7);

Таблиця 3.7 – Колекція favorites (улюблені вправи/сети користувача)

Поле	Тип	Опис
favoriteId	string	Унікальний ID запису (може генеруватись автоматично)
userId	string	ID користувача, якому належить улюблене

itemId	string	ID вправи або сету
itemType	string	Тип елемента: `"exercise"` або `"set" або "recipes"`
addedAt	timestamp	Дата й час додавання в улюблене

– customExercises – вправи, які створив сам користувач (див. табл. 3.8);

Таблиця 3.8 – Колекція customExercises (вправи, створені користувачем)

Поле	Тип	Опис
exerciseId	string	Унікальний ID вправи

Кінець таблиці 3.8

userId	string	ID користувача (автора вправи)
name	string	Назва вправи
description	string	Детальний опис
muscleGroup	string	М'язова група
difficulty	string	Рівень складності
videoUrl	string (optional)	Посилання на демонстраційне відео
createdAt	timestamp	Дата створення

UML-діаграми

Для кращого розуміння роботи системи були побудовані наступні UML-діаграми:

– **Use Case Diagram (діаграма варіантів використання) –** на цій діаграмі (див. рис. 3.1) відображено основні функції, які може виконувати користувач системи: від реєстрації та входу до створення власних вправ і перегляду рекомендацій. Це дозволяє наочно зрозуміти, якими саме сервісами користується кінцевий користувач та як він взаємодіє із системою;

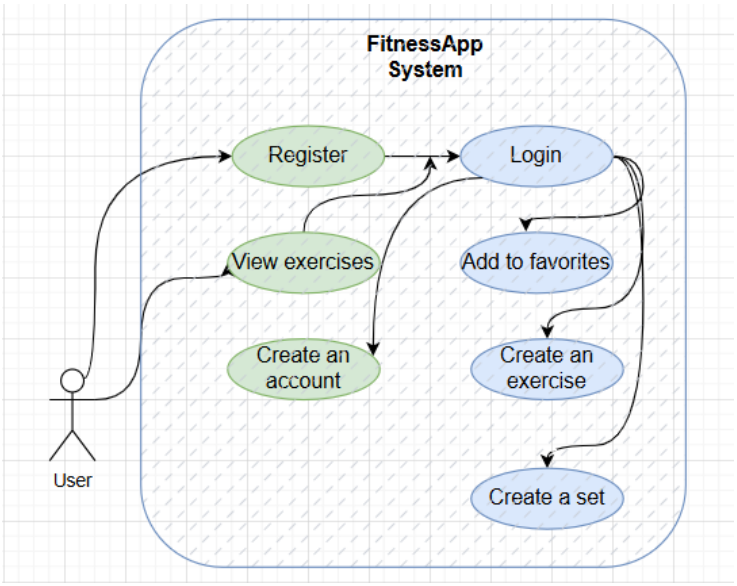

Рисунок 3.1 – Діаграма варіантів використання

– **Activity Diagram (Діаграма активності) –** діаграма активності ілюструє основні кроки взаємодії користувача з системою. Вона дозволяє побачити загальну логіку роботи – від першого входу до взаємодії з персональними рекомендаціями (див. рис. 3.2);

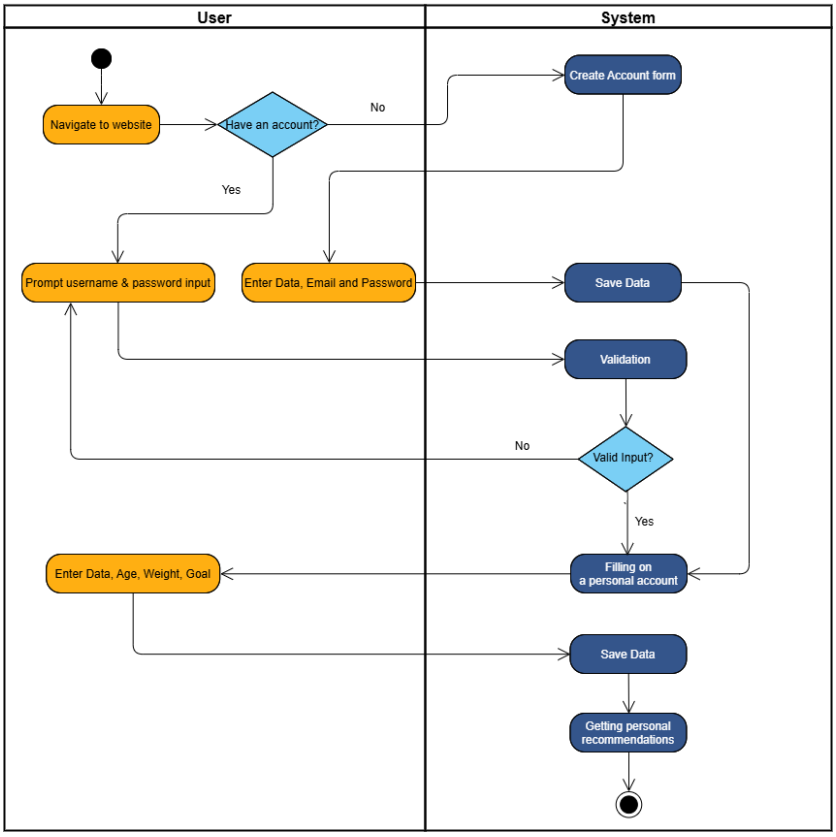

Рисунок 3.2 – Діаграма активності

– **Class Diagram (Діаграма класів) –** діаграма класів показує логічну структуру інформаційної системи. Вона дозволяє побачити, які дані зберігаються у користувача, як реалізовані улюблені елементи, індивідуальні вправи, рекомендації, профіль та базові вправи з API. Завдяки компонентному підходу класи чітко розділені за функціональністю, що спрощує підтримку та масштабування системи;

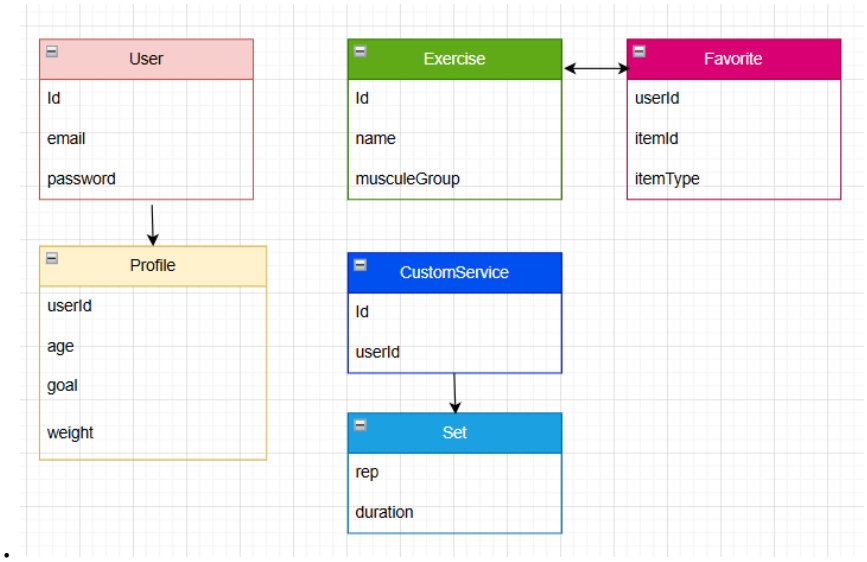

Рисунок 3.3 – Діаграма класів

– **Sequence Diagram (Діаграма послідовностей) –** це зображення (див рис. 3.4) демонструє основну логіку обміну запитами між елементами системи: користувач → інтерфейс → сервер → база даних. Послідовність дозволяє чітко зрозуміти, як проходить авторизація, зберігання інформації, генерація персоналізованого контенту, створення власної вправи та сетів;

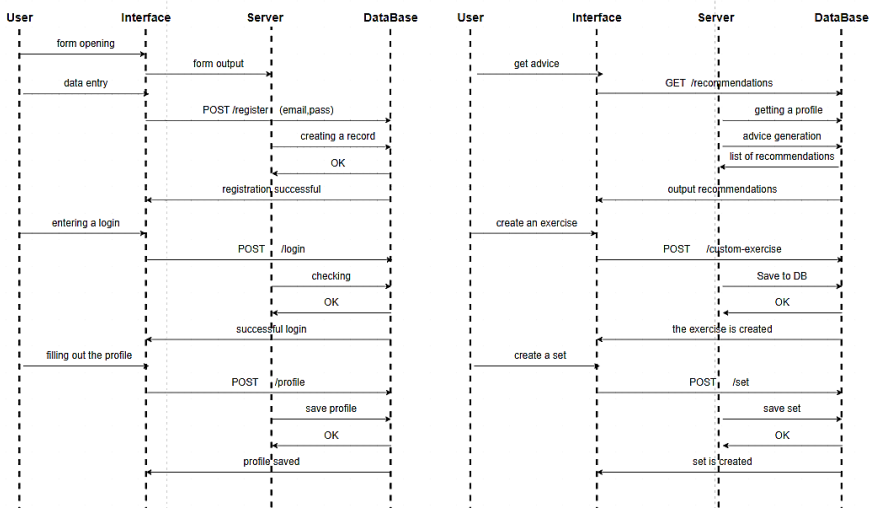

Рисунок 3.4 – Діаграма послідовностей

Дизайн користувацького інтерфейсу.

У цьому пункті представлено візуальну частину інформаційної системи, розробленої з урахуванням зручності використання, логіки навігації та привабливого оформлення для цільової аудиторії. Дизайн виконано у Figma із застосуванням принципів сучасного UI/UX-дизайну.

На даному рисунку представлено макет (див. рис. 3.5) сторінки авторизації, а саме login форми.

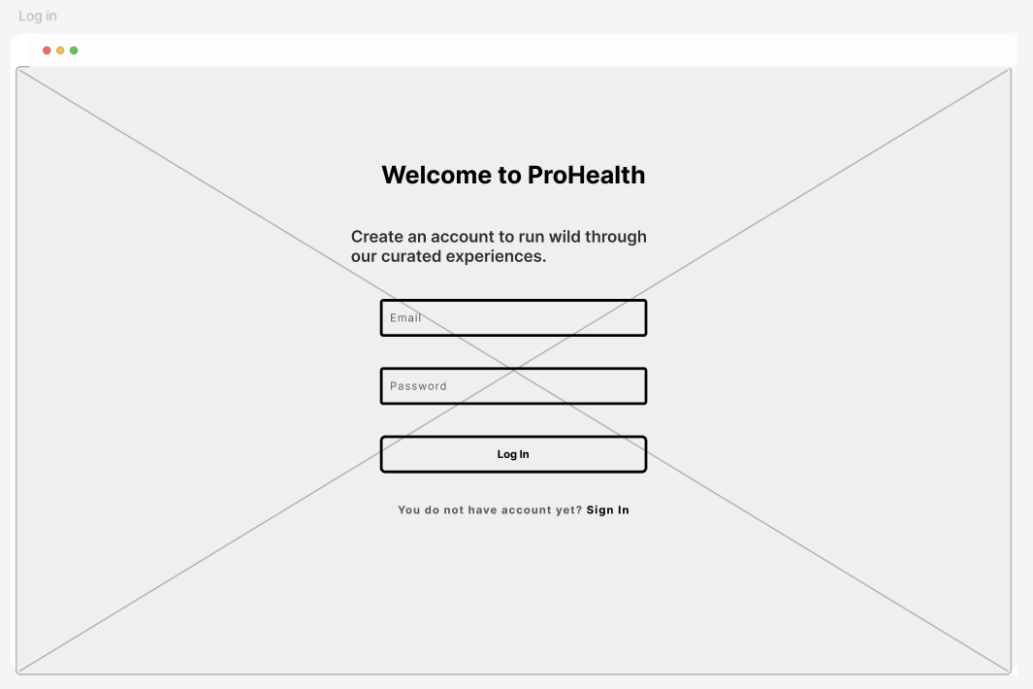

Рисунок 3.5 – Макет сторінки входу в систему

Якщо користувач ще не був зареєстрований, то натискаючи на посилання Sign in, автоматично буде направлений на сторінку реєстрації (див. рис 3.6).

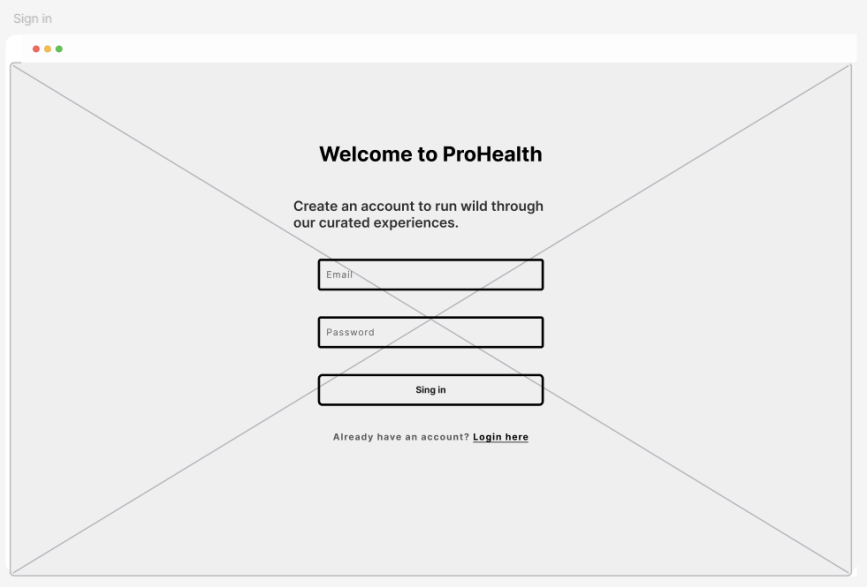

Рисунок 3.6 – Макет сторінки реєстрації

Після успішної реєстрації користувач перенапралвяється на головну сторінку сайту (див. рис. 3.7).

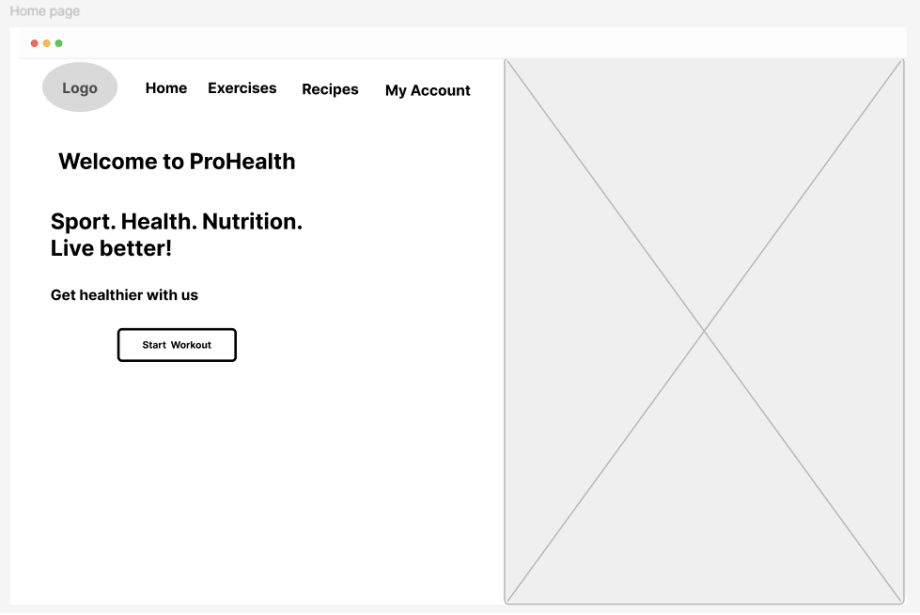

Рисунок 3.7 – Макет головної сторінки сайту

На сторінці вправ користувач бачить зручний та інтуїтивно зрозумілий інтерфейс, у якому можна ввести назву вправи в пошуковому рядку або обрати вже згруповані вправи за певними групами м'язів, наприклад, різноманітні вправи на дельтоподібні м'язи (плечі) (див. рис. 3.8).

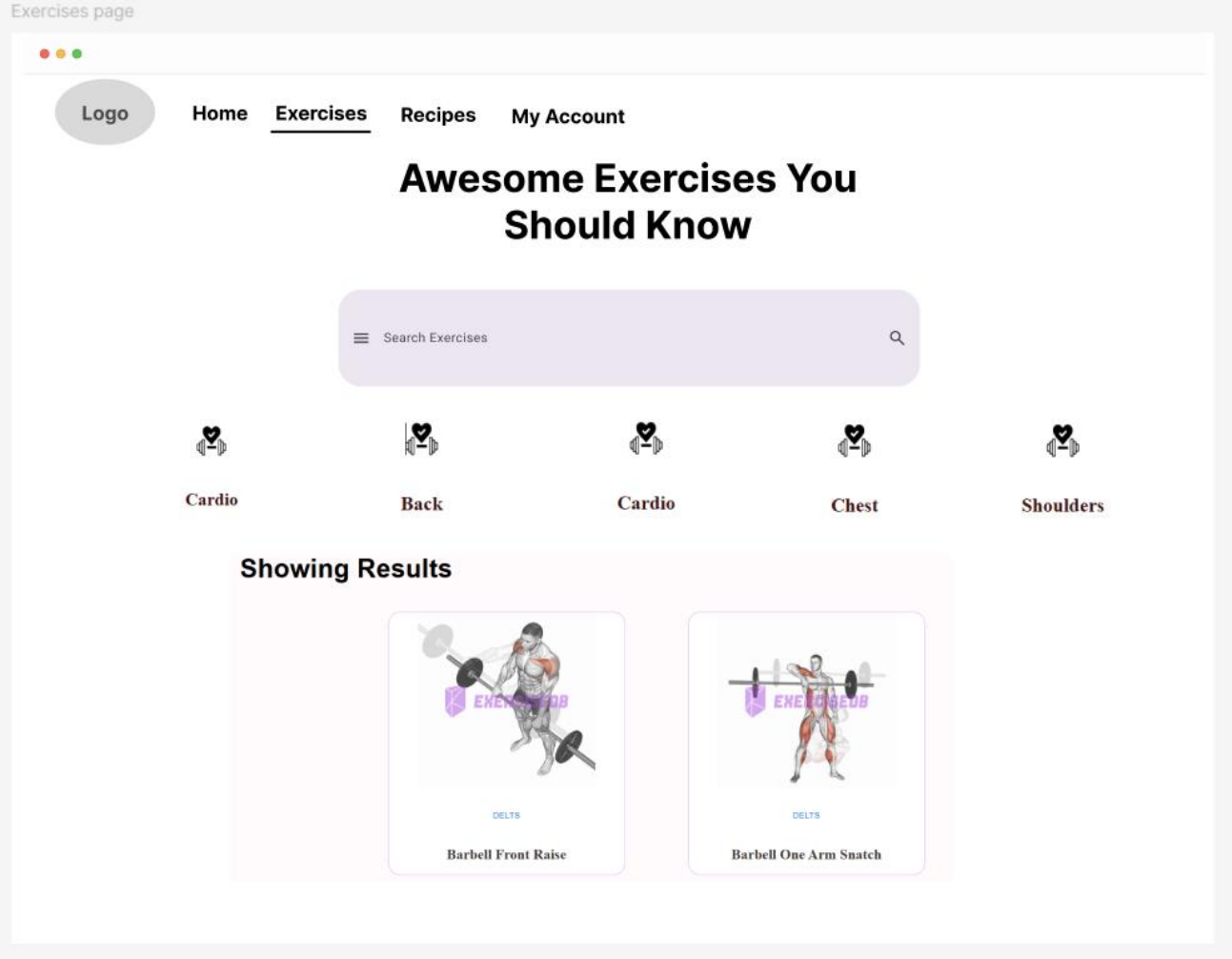

Рисунок 3.9 – Макет сторінки з фізичними вправами

Після того як користувач зацікавився якоюсь вправою, натискаючи на неї, він переходить на сторінку, де можна побачити детальний опис про цю чи іншу вправу, а також може подивитись правильність виконання, безпосередньо, по

відео з платформи YouTube. Також користувачу буде запропоновано схожі вправи на ту групу м’язів яку він обрав. Так само буде запропоновано схожі вправи із схожим спортивним обладнанням. До того ж, користувачі зможуть додавати вправи в уподобане, для того щоб потім швидко знайти і тренуватись по відео професійних людей з правильною технікою.

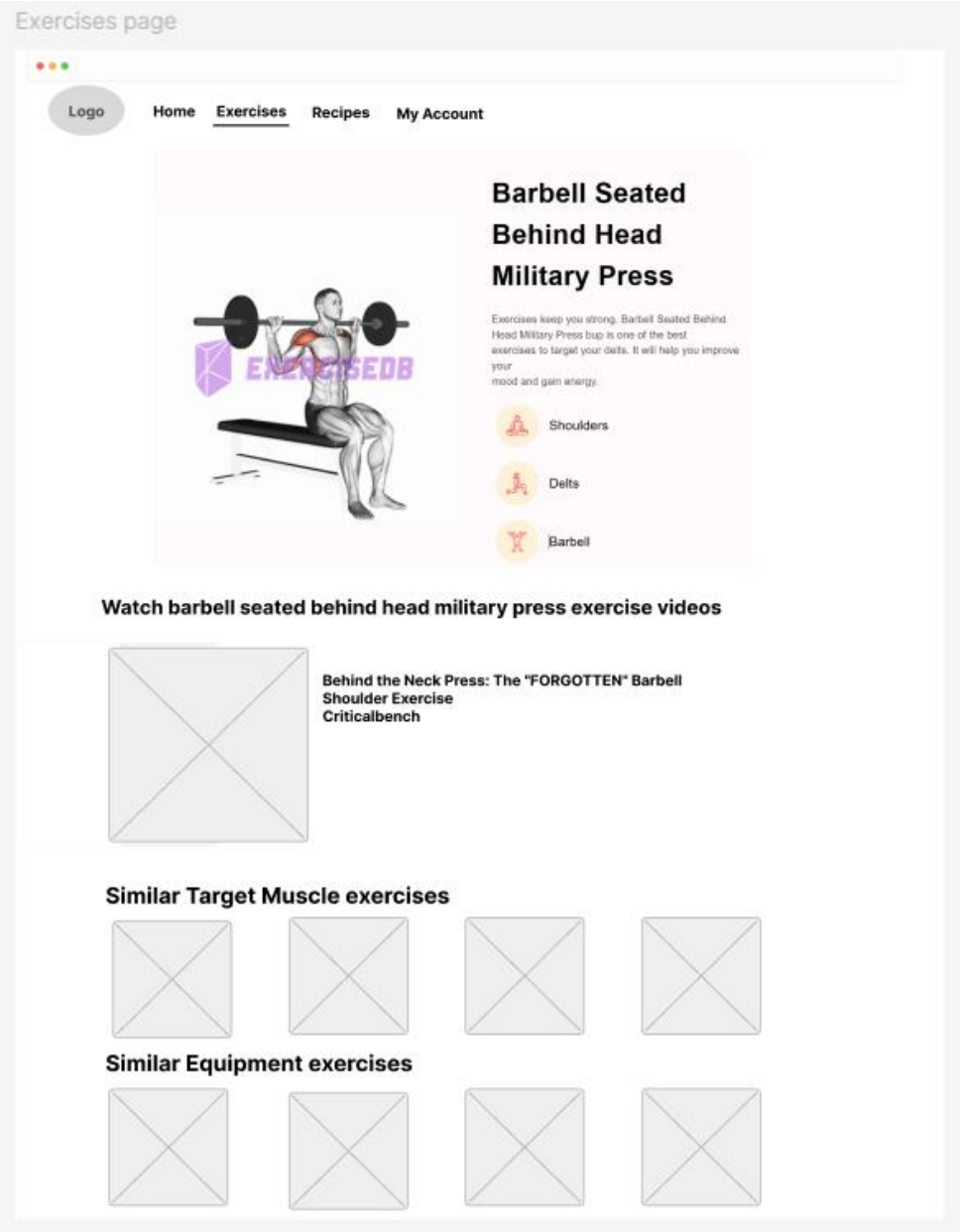

Рисунок 3.10 – Макет сторінки до виконання певної вправи

Тепер перейдемо до рецептів. Сторінка з рецептами буде виглядати таким чином (див. рис. 3.11). На цьому макеті можна відразу побачити скільки часу займе приготування страви.

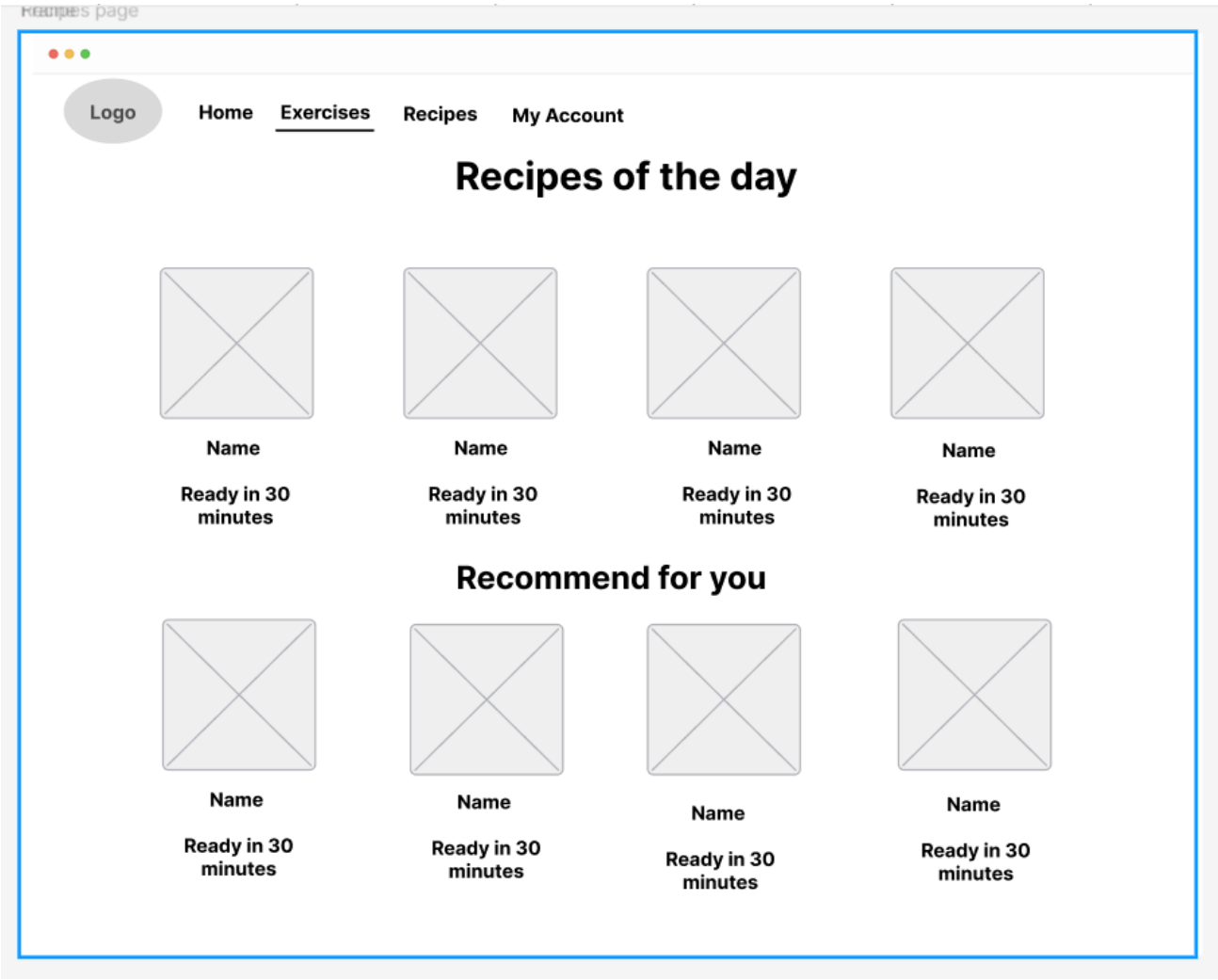

Рисунок 3.11 – Макет сторінки з корисними та поживними рецептами

На цій сторінці користувач може подивитися інгредієнти, вміст кілокалорій, цукру, жиру та білку та детальний опис приготування обраного рецепту. А також додати в «Улюблене» натиснувши на фігуру серця (див. рис. 3.12).

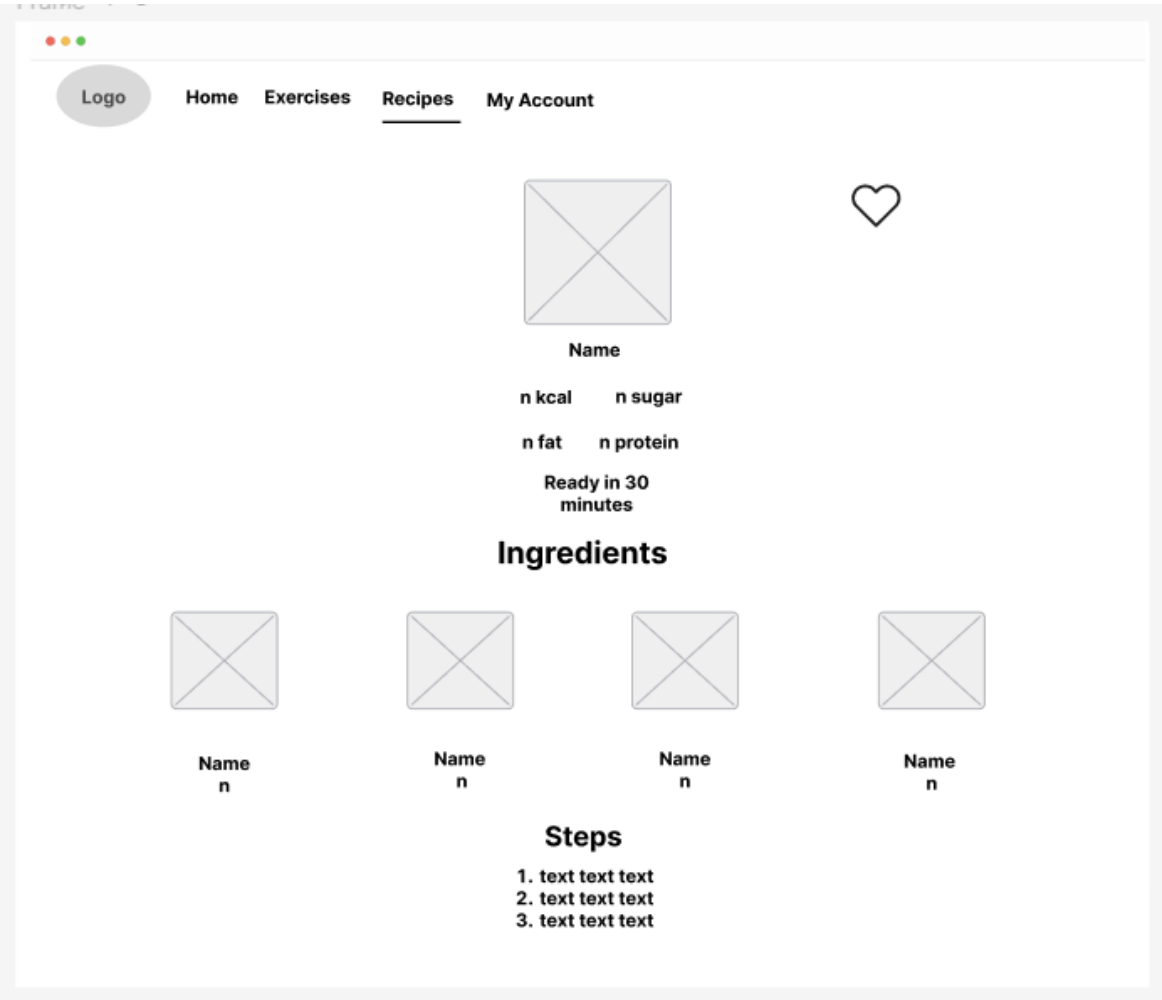

Рисунок 3.12 – Макет сторінки певного рецепту з кроками приготування

У особистому кабінеті (див рис. 3.13) користувач може редагувати власний профіль (змінювати ім’я, електрону пошту, пароль і тд.). До того ж, там розміщені посилання на форми (сторінки) створення сетів з вже доступних вправ, або зі створених особисто. Так само може перейти на сторінку уподобаних вправ або рецептів, які будуть зберігатись там, до поки користувач не видалить їх особисто.

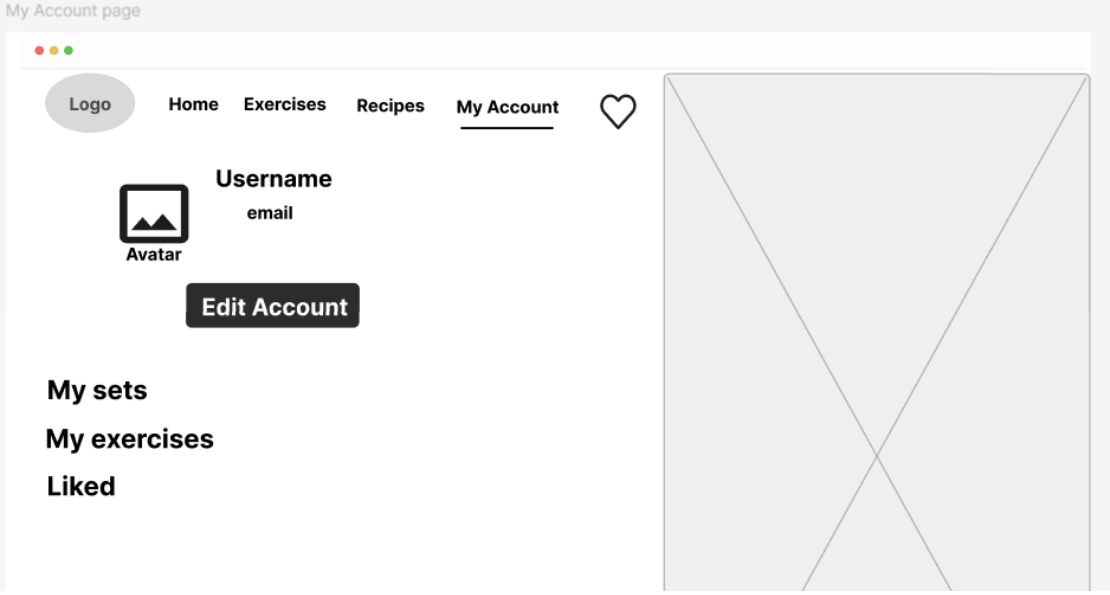

Рисунок 3.13 – Макет сторінки особистого кабінету (My Account)

На сторінці з обраними вправами та рецептами, можна побачити назву та фото кожної вправи і рецепту. Так само можна видалити з обраного, те що користувачеві більше не до вподоби.

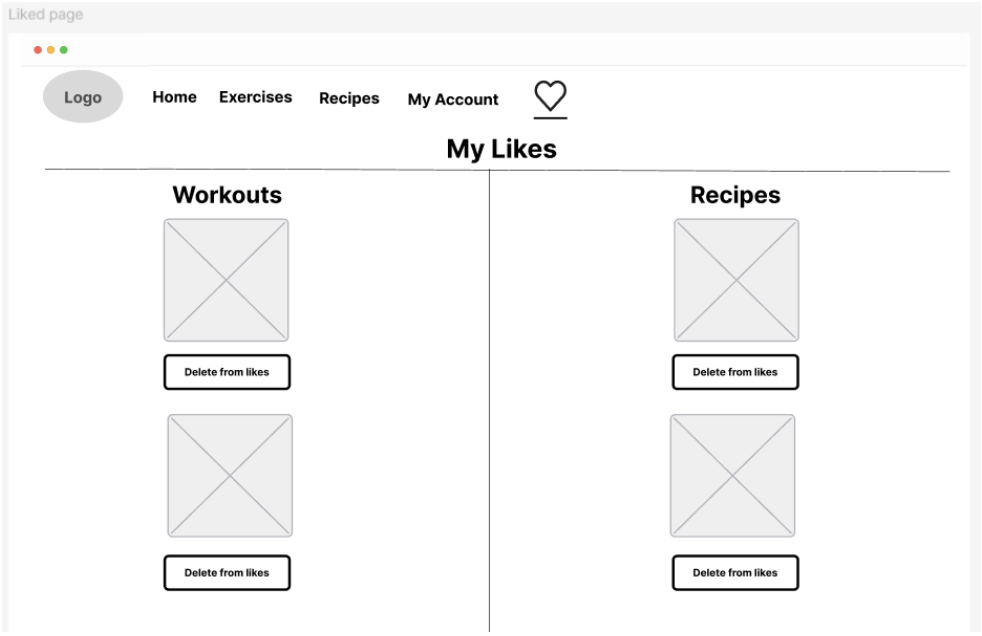

Рисунок 3.14 – Макет сторінки «Уподобане»

На сторінці з сетами можна створити сласний сет додаючи вправи (із випадаючого списку / пошуку за назвою) (див. рис. 3.15).

При натисканні на кнопку "View" користувач переходить на сторінку з обраним сетом, де він може редагувати назву, або опис із кнопкою збереження. Нижче користувач може побачити створений сет, а також є функції зміни положення та додання додаткової вправи. Присутня кнопка початку вправи, де буде відраховуватись час. Також за потреби можна видалити сет (див. рис. 3. 16).

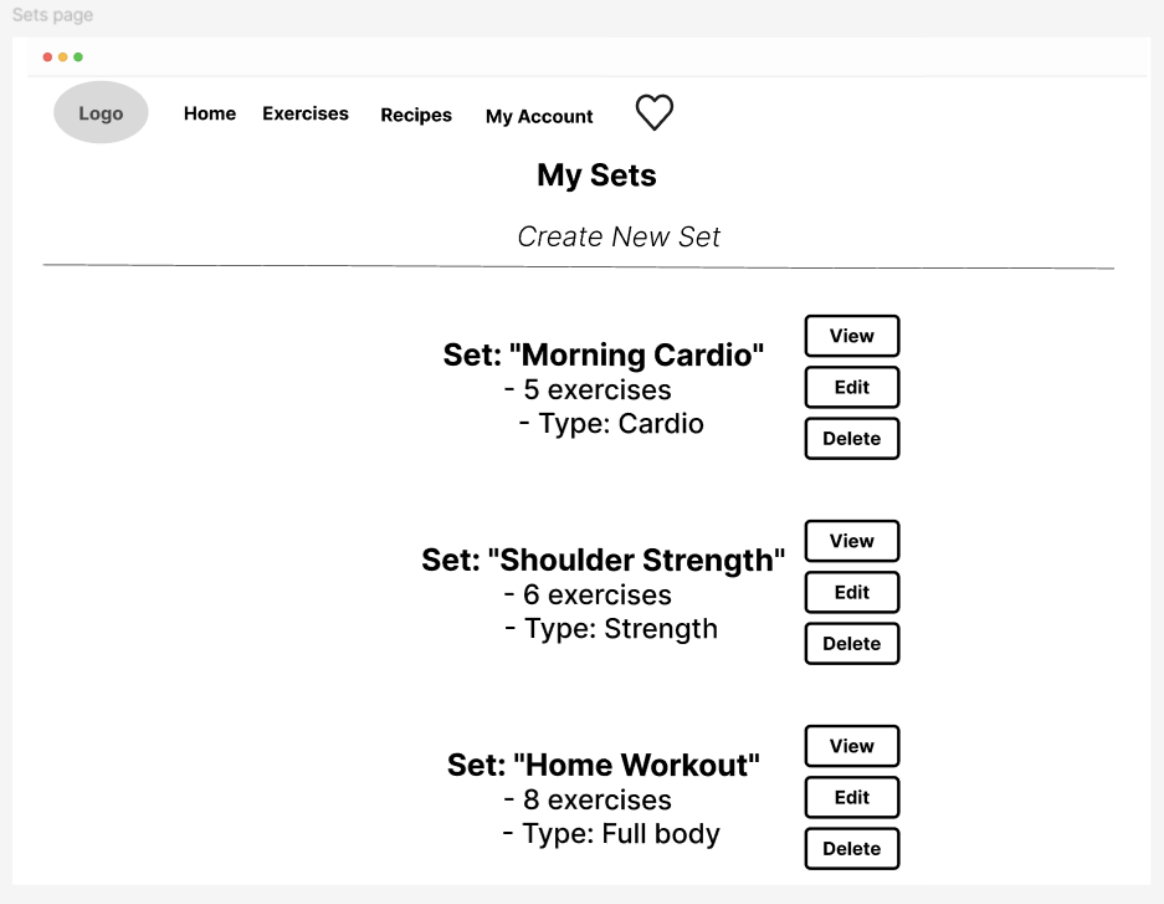

Рисунок 3.15 – Макет сторінки «Мої сети»

При натисканні на посилання для створення власного сету «Create New Set» користувач зможе заповнити таку форму (див. табл. 3.10). Після збереження, сет

з’являється у вкладці My Sets, а обрані вправи всередині нього можуть мати коротке прев’ю.

Таблиця 3.10 – Форма для створення власного сету

Поле	Тип
Set Name	Текстове поле (назва сету, наприклад: Full Body Workout)
Category	Випадаючий список (Cardio Set, Strength Set, Flexibility Set…)
Target Area	Вибір м'язових груп (Chest, Legs, Back, Full Body…)
Select Exercises	Список з чекбоксами або drag-and-drop із доданих вправ
Total Duration (min)	Числове поле
Notes / Description	Текстове поле (опис мети сету, поради тощо)
Upload Cover Photo	Поле для завантаження зображення (формати: .jpg, .png)
Save Set	Кнопка збереження

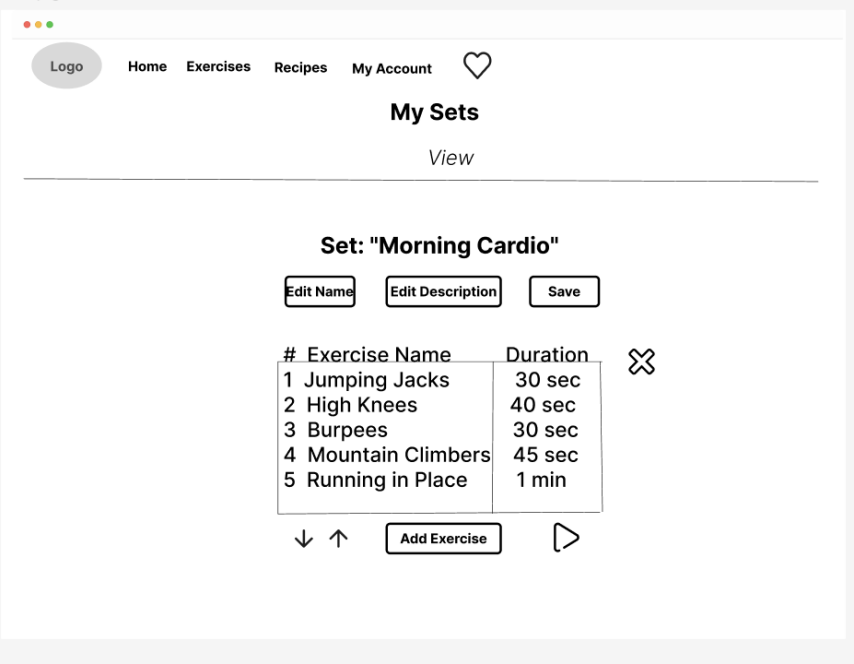

Рисунок 3.16 – Макет сторінки огляду обраного сету

Також користувач зможе додати власні вправи для тренувань, з можливістю їх редагування або видалення. Також у користувача буде можливість додати вправу до набору (сету).

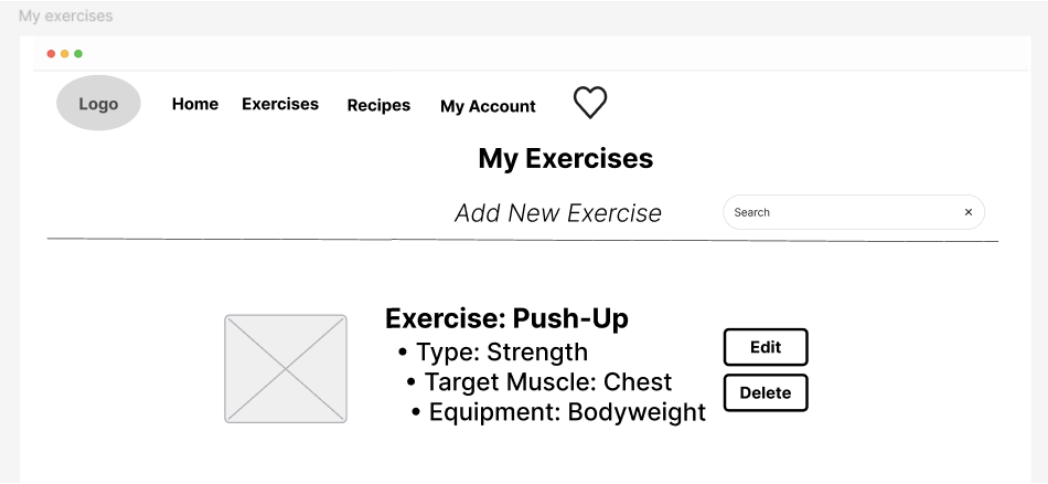

Рисунок 3.17 – Макет сторінки «Мої вправи»

При натисканні на посилання для додавання власної вправи «Add New Exercise» користувач зможе заповнити таку форму:

Таблиця 3.11 – Форма для додавання власної вправи

Поле	Тип
Exercise Name	Текстове поле
Type	Випадаючий список (Cardio, Strength, Flexibility)
Target Muscle	Вибір з м'язів (наприклад: chest, legs, abs…)
Equipment	Текстове поле або вибір (Bodyweight, Dumbbell…)
Duration / Reps / Sets	Числові поля
Notes / Description	Текстове поле
Upload photo	Поле для завантаження зображення (формати: .jpg, .png)
Save	Кнопка збереження

Дизайн інформаційної системи виконаний у сучасному мінімалістичному

стилі з акцентом на зручність користування. Інтерфейс інтуїтивний та структурований, включає основні сторінки: Головна, Вправи, Рецепти, Авторизація, Мій акаунт, Мої вправи, Мої сети, Уподобане. Користувач може створювати власні вправи й набори вправ, додавати описи та зображення.

Висновки до розділу 3

У результаті реалізації інформаційної системи для оцінки здоров’я та фізичних навантажень було досягнуто основних функціональних та структурних цілей, поставлених на етапі проєктування. Система поєднує модулі обліку фізичних параметрів користувача, генерації персональних рекомендацій, управління власними вправами, створення наборів (сетів) та перегляду корисних рецептів.

Розроблена інформаційна система дозволяє користувачам:

– проходити реєстрацію та створювати профіль зі збереженням основних фізіологічних характеристик;

– переглядати, фільтрувати та зберігати вправи у «Улюблене», а також створювати власні вправи з описом, відео та цільовою групою м’язів;

– формувати набори вправ (сети) з вибраних або власних елементів та запускати тренування з візуальним таймером;

– ознайомлюватися з рецептами здорових страв, відбираючи їх відповідно до поставленої мети (схуднення, набір маси, підтримка форми);

– керувати власним обліковим записом, редагувати дані, переглядати активність.

Система функціонує на основі архітектури «клієнт–сервер», з обміном даними через REST API та зберіганням усієї інформації у гнучкій NoSQL-базі

Firebase Firestore. Це забезпечує масштабованість, високу швидкодію та адаптивність до зростання кількості користувачів.

Проєктні рішення підтвердили свою доцільність. Усі заявлені функції реалізовано повністю або частково (з можливістю подальшого розширення), структура бази даних оптимізована під персоналізовану взаємодію, а навігація та дизайн адаптовані для зручного використання навіть недосвідченими користувачами.

Таким чином, результати практики підтверджують ефективність реалізованої концепції інформаційної системи, що здатна виконувати роль цифрового помічника у підтримці здорового способу життя. Отримані результати створюють міцну основу для подальшого розвитку, тестування та впровадження проєкту в реальні умови використання.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ФУНКЦІОНАЛУ ІНФОРМАЦІНОЇ СИСТЕМИ ОЦІНКИ ЗДОРОВ'Я ТА ФІЗИЧНИХ НАВАНТАЖЕНЬ

4.1 Опис програмної реалізації

Розроблена інформаційна система реалізована із використанням сучасних технологій фронтенду та хмарного бекенду, що забезпечує її інтерактивність, масштабованість і зручність для користувача. Інтерфейс побудований на основі бібліотеки React, що дозволяє компонентно структурувати сторінки й ефективно оновлювати об'єктну модель документу DOM (Document Object Model).

Для автентифікації користувача та зберігання даних застосовано платформу Firebase. Firebase Authentication дозволяє користувачам проходити реєстрацію та авторизацію за допомогою email і пароля. Для обробки введених даних на сторінці реєстрації реалізовано перевірку валідності email-адреси, а також наявності великої літери, цифри й мінімальної довжини у паролі. При реєстрації або вході обробляються можливі помилки, а в разі успіху користувача перенаправляє на відповідну сторінку.

Код для реєстрації:

```
    const handleRegister = async (e) => {
      e.preventDefault();
      setError("");

      const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!emailRegex.test(email)) {
        setError("Please enter a valid email address.");
        return;
      }
      if (!isPasswordValid(password)) {
        setError("Password must be at least 6 characters long,
include an uppercase letter and a number.");
        return;
      }
      try {
```

```
        await createUserWithEmailAndPassword(auth, email,
password);
        alert("Registration successful!");
        navigate("/login");
      } catch (err) {
        setError(err.message);
      }
    };
```

Код логіну:

```
    const handleLogin = async (e) => {
      e.preventDefault();
      setError("");
      setSuccess("");
      const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!emailRegex.test(email)) {
        setError("Please enter a valid email address.");
        return;
      }
      if (!isPasswordValid(password)) {
        setError("Password must be at least 6 characters long,
include an uppercase letter and a number.");
        return;
      }
      try {
        await signInWithEmailAndPassword(auth, email, password);
        setSuccess("Login successful!");
        navigate("/");
      } catch (err) {
        setError(err.message);
      }
    };
```

Більш детально цей фрагмент коду та наступні можна розглянути в **Додатку А.**

Усі особисті дані (такі як ім'я, дата народження, email, вага та зріст) зберігаються у Firestore – гнучкій, масштабованій NoSQL базі даних. Після реєстрації користувача його дані додаються до колекції "users". Firebase також використовується для зберігання користувацьких вправ, сетів та улюблених елементів. Ініціалізація бази даних виглядає так:

```
    import { initializeApp } from "firebase/app";
```

```
    import { getAuth } from "firebase/auth";
    import { getFirestore } from "firebase/firestore";

    const firebaseConfig = {
      apiKey: "AIzaSyBpCiWVZTvqYxrcgX7frevI9z7WM9rkfxo",
      authDomain: "fitnessapp-e09eb.firebaseapp.com",
      projectId: "fitnessapp-e09eb",
      storageBucket: "fitnessapp-e09eb.appspot.com",
      messagingSenderId: "218069016353",
      appId: "1:218069016353:web:28d5d43a0d72e740502860"
    };

    const app = initializeApp(firebaseConfig);
    export const auth = getAuth(app);
export const db = getFirestore(app);
```

Сторінка "My Account" відображається лише після успішної авторизації користувача. Вона містить форму з можливістю редагування особистих даних: імені, дати народження, електронної пошти, ваги та зросту. При зміні значень у формах дані автоматично оновлюються у базі Firestore через функцію updateDoc.

Код збереження змін виглядає так:

```
    const handleSave = async () => {
      const userRef = doc(db, "users", user.uid);
      await updateDoc(userRef, {
        name,
        birthday,
        email,
        weight,
        height
      });
      toast.success("Дані оновлено!");
};
```

Окрім профілю користувача, реалізовано можливість створення власних вправ. Користувач вводить назву, опис, групу м’язів і кількість повторень. Після натискання кнопки «Зберегти», об’єкт додається до колекції "customExercises" у Firestore:

```
    await addDoc(collection(db, "customExercises"), {
      userId: user.uid,
      title,
      description,
      muscleGroup,
      repetitions,
      createdAt: serverTimestamp()
    });
```

Подібно працює функціонал створення сетів. Користувач обирає з наявних вправ і формує набір під певною назвою. Сети зберігаються в окремій колекції:

```
    await addDoc(collection(db, "sets"), {
      userId: user.uid,
      name,
      exercises: selectedExercises,
      createdAt: serverTimestamp()
    });
```

Для кожного сету додатково реалізовано таймер, який дозволяє користувачу відстежувати час виконання вправи. Таймер запускається зі сторінки перегляду сету та працює локально у межах компонента.

У розділі вправ реалізовано сторінку, що отримує дані з API ExerciseDB. Користувач може фільтрувати вправи за частинами тіла, групами м'язів та типом обладнання. Детальна інформація про вправу відкривається окремою сторінкою. Там відображаються gif-анімація, опис, група м'язів, схожі вправи й відео з YouTube.

Користувач має змогу додати вправу або рецепт до улюбленого, натиснувши кнопку з іконкою лайку. Дані зберігаються в колекції "favorites":

```
    await addDoc(collection(db, "favorites"), {
      userId: user.uid,
      itemId: item.id,
      itemType: "exercise" | "recipe",
```

```
        savedAt: serverTimestamp()
});
```

Розділ з рецептами отримує інформацію з Spoonacular API. На головній сторінці виводяться рецепти дня, швидкі рецепти, жарт дня та факт дня. Кожен рецепт можна переглянути повністю: виводяться інгредієнти, поживна цінність (калорії, жири, білки, вуглеводи), кроки приготування. Усі ці елементи винесено в окремі компоненти – ItemCard, Ingredients, Steps.

Таким чином, додаток об'єднує особистий обліковий запис користувача з можливістю створювати, зберігати та керувати контентом, використовуючи Firebase та зовнішні API. Кожен розділ має логічно завершену функціональність, а дані зберігаються в структурованому вигляді в Firestore.

4.2 Керівництво користувача

У цьому підрозділі наведено покрокову інструкцію користування інформаційною системою. Зазначені етапи охоплюють роботу з основними функціями сайту: реєстрацію, вхід, редагування профілю, перегляд та створення вправ і сетів, роботу з рецептами та додавання елементів до улюбленого.

Після відкриття сайту користувач потрапляє на головну сторінку, де може ознайомитися з рекомендованими рецептами та вправами. У верхньому меню розміщено посилання на сторінку реєстрації (Register).

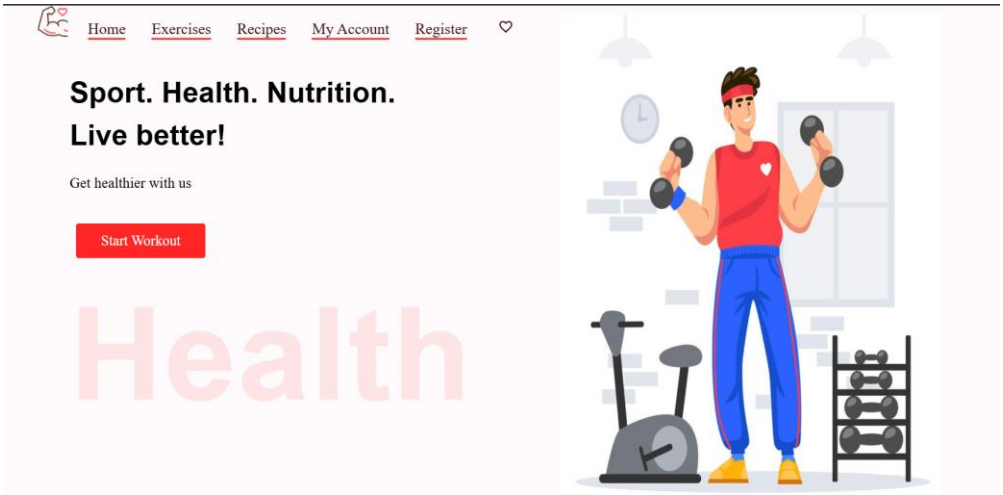

Рисунок 4.1 – Головна сторінка інформаційної системи

Реєстрація відбувається шляхом введення email та паролю. Після успішної реєстрації користувач перенаправляється на сторінку логіну. Пароль має відповідати вимогам: не менше 6 символів, одна велика літера та хоча б одна цифра. При некоректному введенні з'являється відповідне повідомлення про помилку.

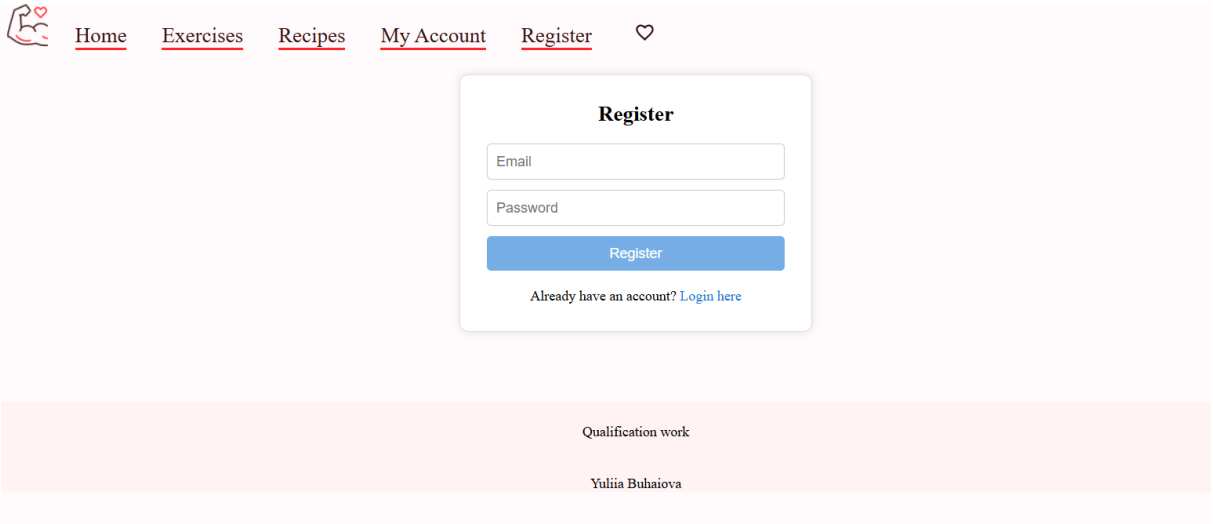

Рисунок 4.2 – Форма реєстрації

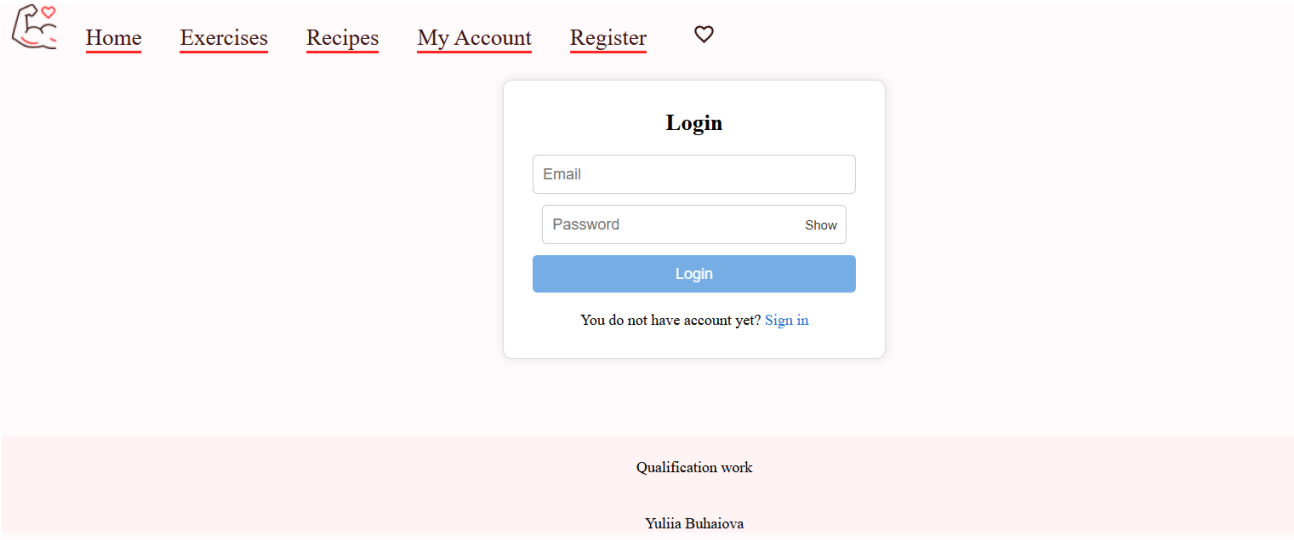

Рисунок 4.3 – Форма входу після успішної реєстрації

Після входу користувач бачить кнопку переходу на сторінку "My Account", де відображається форма з полями: ім'я, email, вага, зріст. Усі дані можна редагувати і зберегти натисканням на відповідну кнопку. Успішне збереження супроводжується повідомленням.

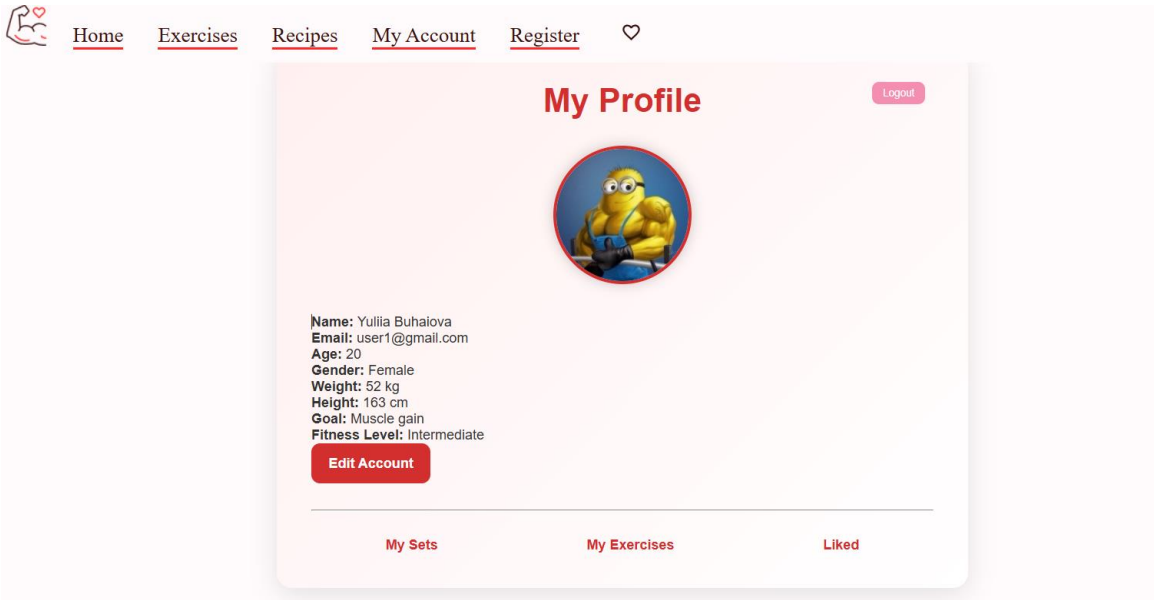

Рисунок 4.4 – Сторінка створення власного аккаунту

На сторінці "Вправи" користувач може скористатись пошуком для відбору вправ за ключовими словами. Також реалізований фільтр за частинами тіла. Всі знайдені вправи відображаються у вигляді інтерактивних карток. При натисканні на одну з вправ відкривається детальна сторінка, де відображається назва вправи, gif-анімація, цільова м’язова група, подібні вправи та відео з YouTube. Поруч із назвою вправи розміщена кнопка додавання до улюбленого – вона позначається іконкою серця.

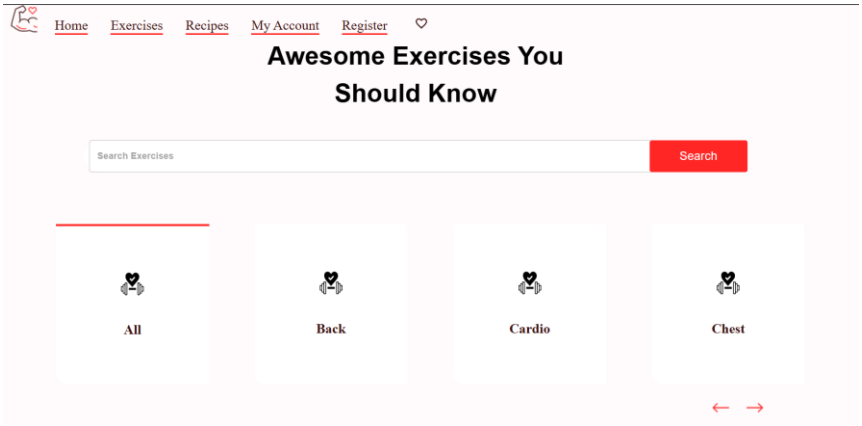

Рисунок 4.5 – Скріншот пошуку вправ і фільтрації

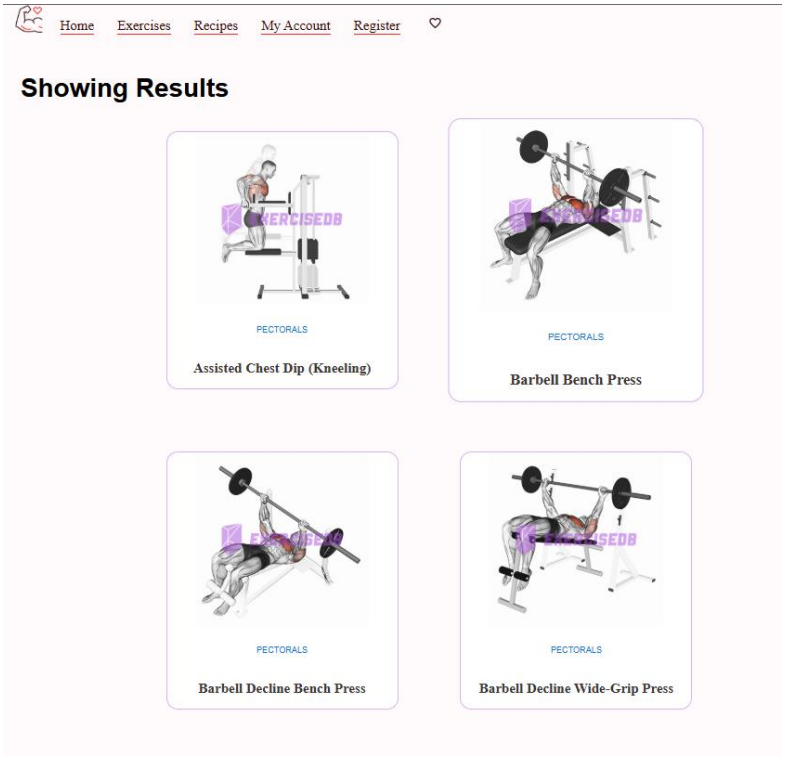

Рисунок 4.6 – Скріншот картки вправи

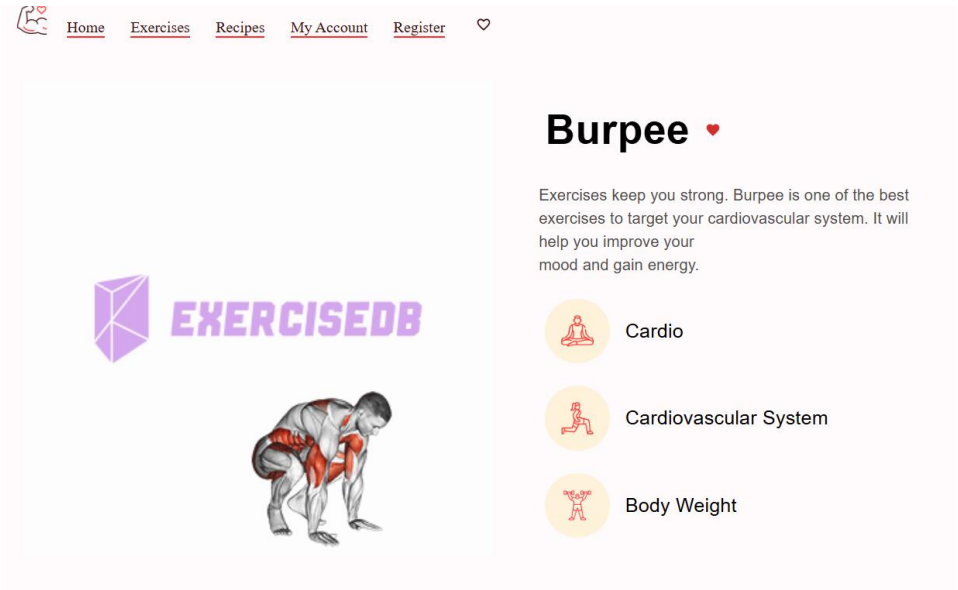

Рисунок 4.7 – Скріншот детальної сторінки вправи з відео та кнопкою лайку

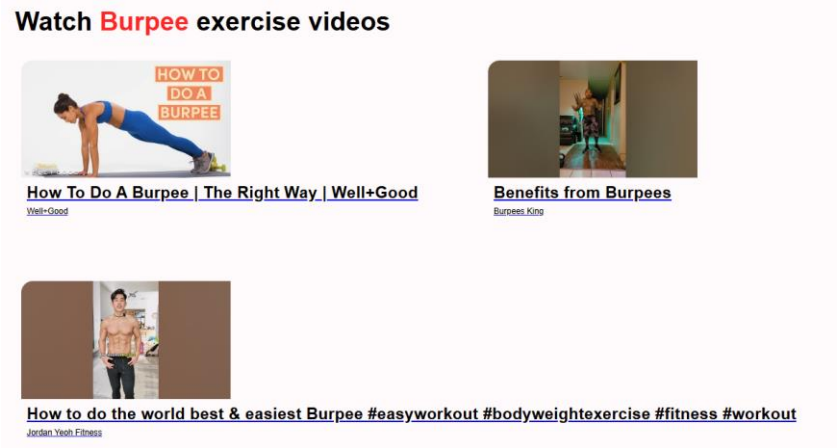

Рисунок 4.8 – Скріншот відео матеріалів для вправи

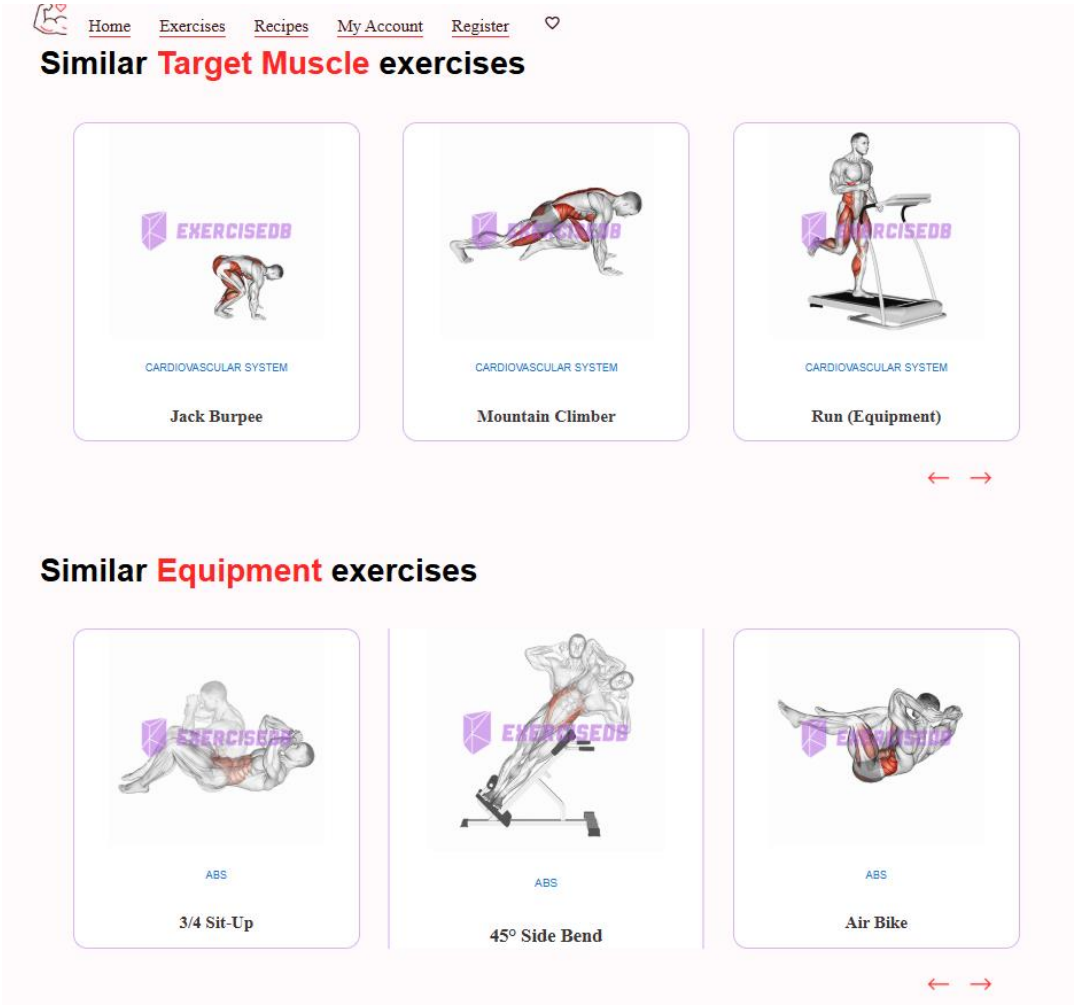

Рисунок 4.9 – Скріншот подібних вправ за зоною м'язів та обладнання

На сторінці "Мої вправи" що знаходиться в особистому акаунті користувач має змогу створити власну вправу, заповнивши відповідну форму (назва, опис, м’язова група, кількість повторень). Після натискання кнопки "Зберегти" створена вправа додається до бази даних та відображається в особистому списку.

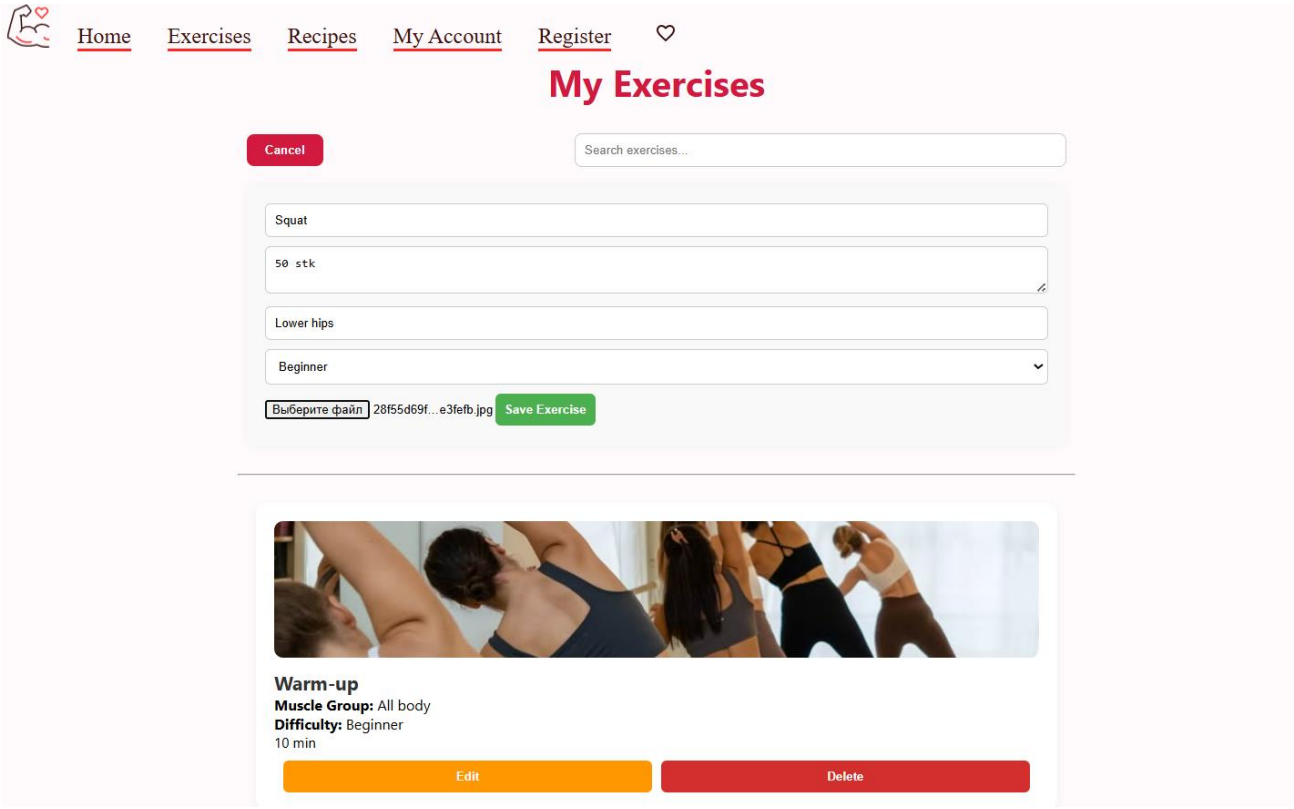

Рисунок 4.10 – Скріншот форми створення нової вправи

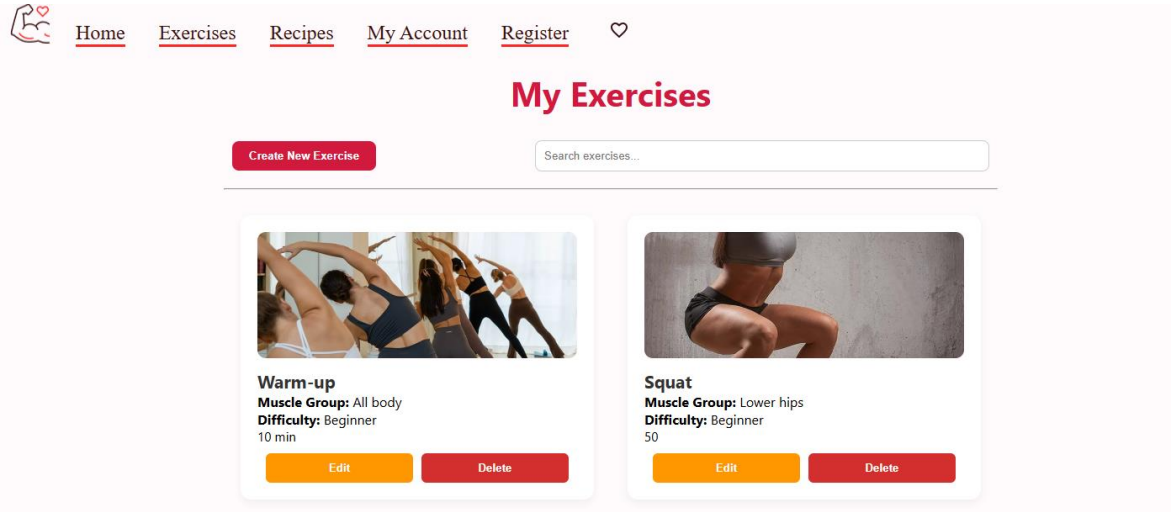

Рисунок 4.11 – Скріншот списку створених користувацьких вправ

У розділі "Мої сети" користувач може створити набір вправ. Для цього необхідно ввести назву сету та вибрати вправи зі списку доступних. Збережений сет буде доступний на цій же сторінці, а при його відкритті запускається вбудований таймер, що допомагає контролювати час виконання вправ.

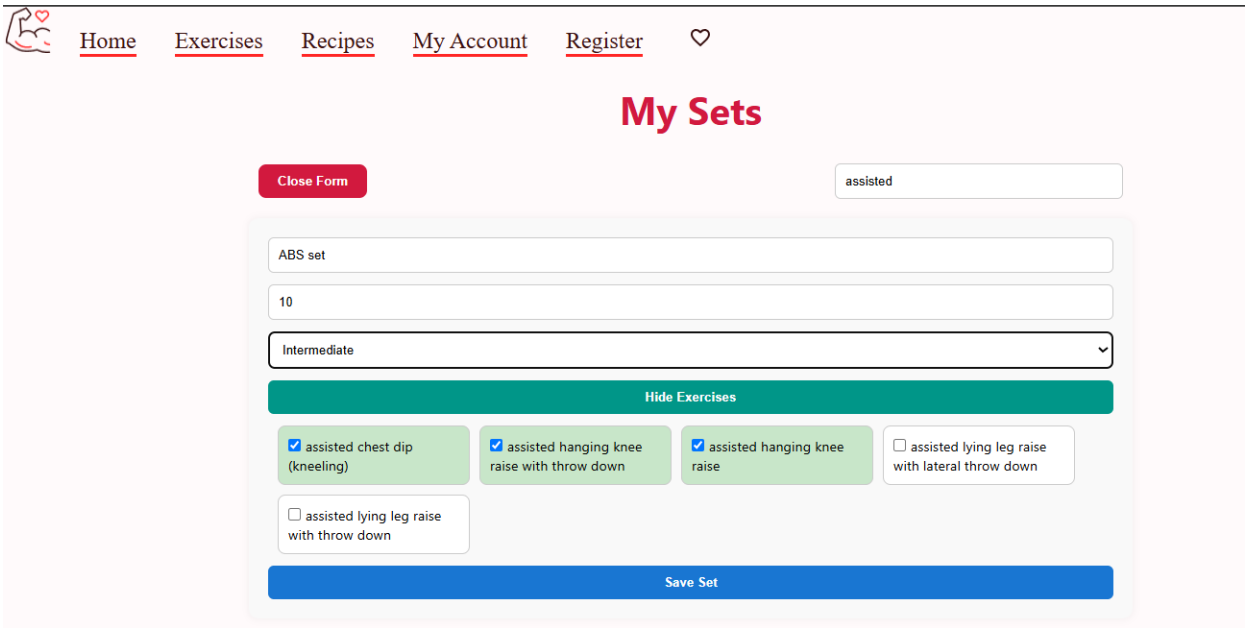

Рисунок 4.12 – Скріншот створення нового сету

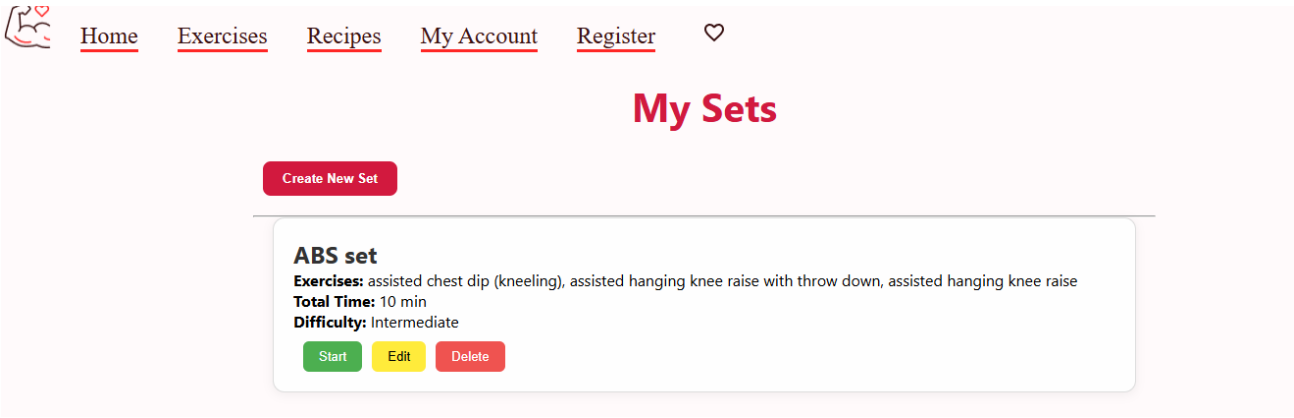

Рисунок 4.13 – Скріншот створеного сету для кору м'язів животу

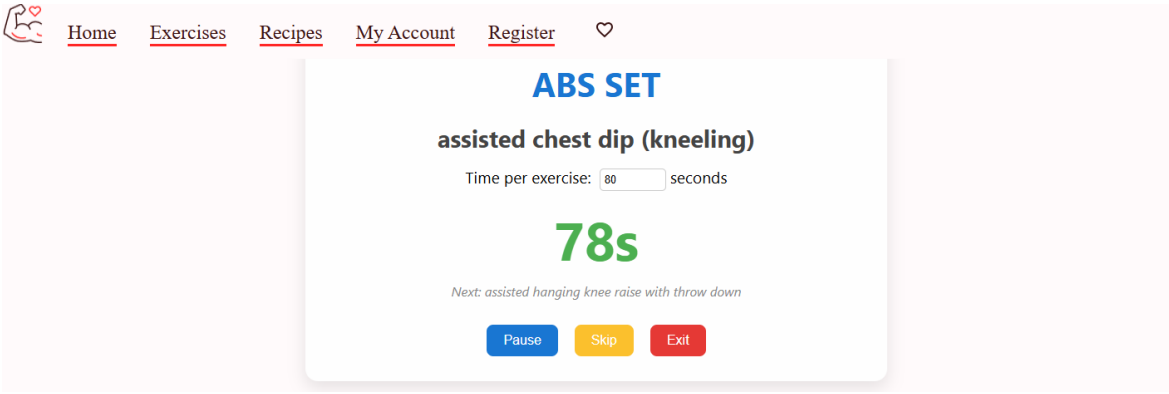

Рисунок 4.14 – Скріншот працюючого таймера

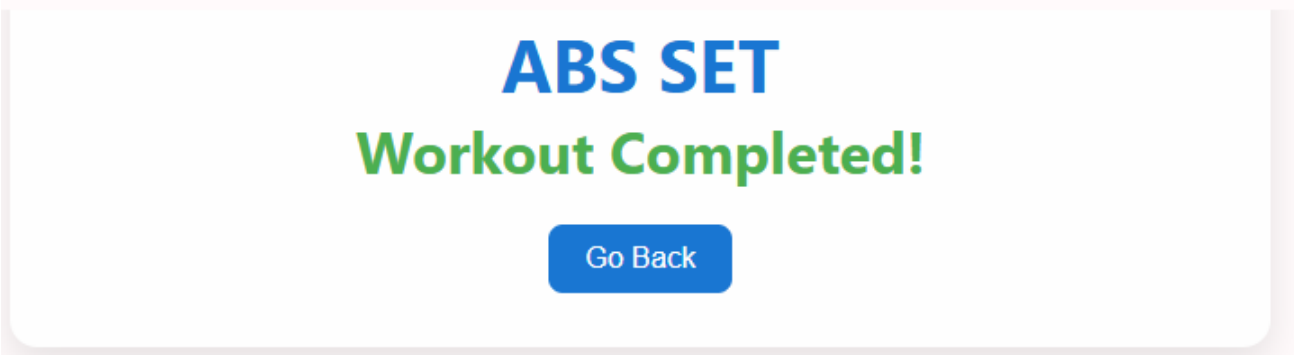

Рисунок 4.15 – Скріншот успішного закінчення вправи

Розділ рецептів доступний з головного меню. На сторінці відображаються

блоки "Recipes of the Day", "Quick Recipes", жарт дня та цікавий факт про їжу. Кожен рецепт представлений у вигляді картки із зображенням, назвою та часом приготування. При натисканні відкривається детальна інформація про страву: інгредієнти, кроки приготування, нутрієнти (калорії, білки, жири, вуглеводи). Рецепт також можна додати до улюбленого натисканням кнопки лайку.

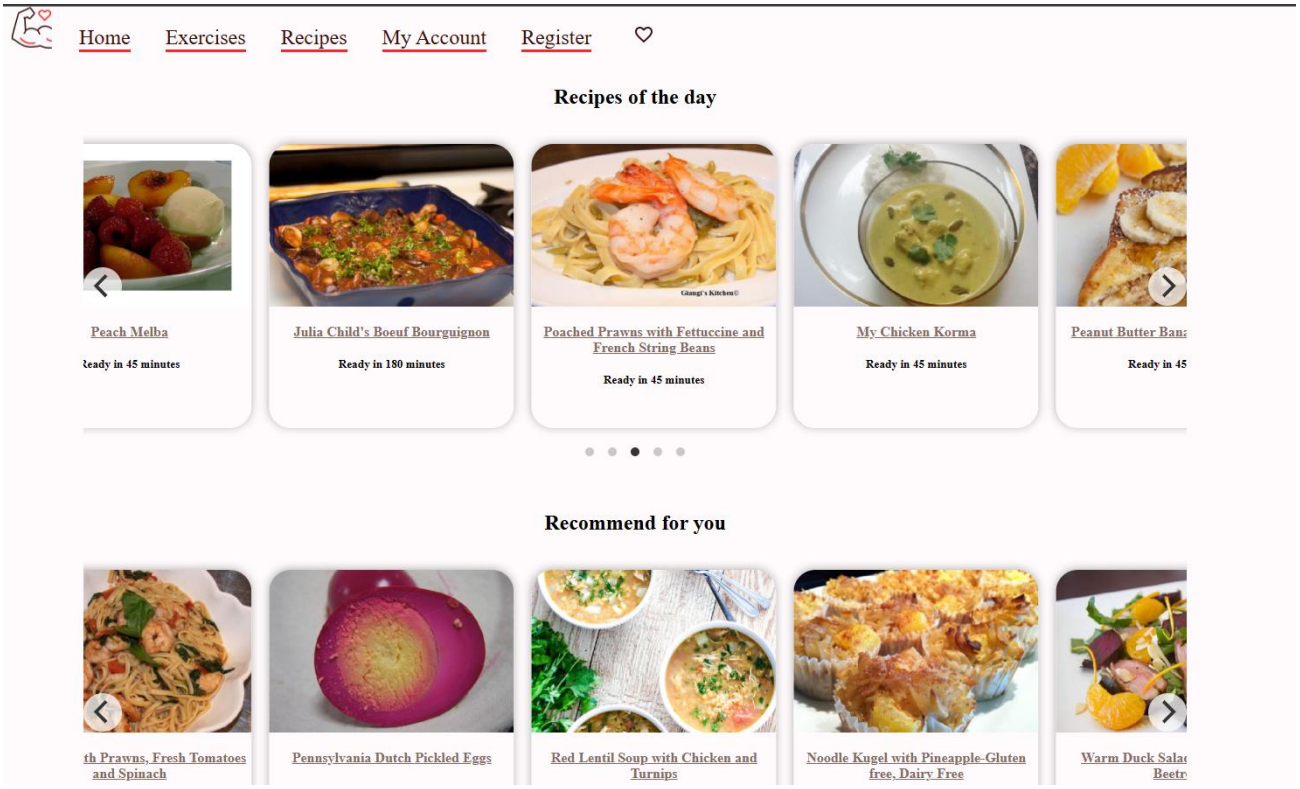

Рисунок 4.16 – Скріншот головної сторінки рецептів

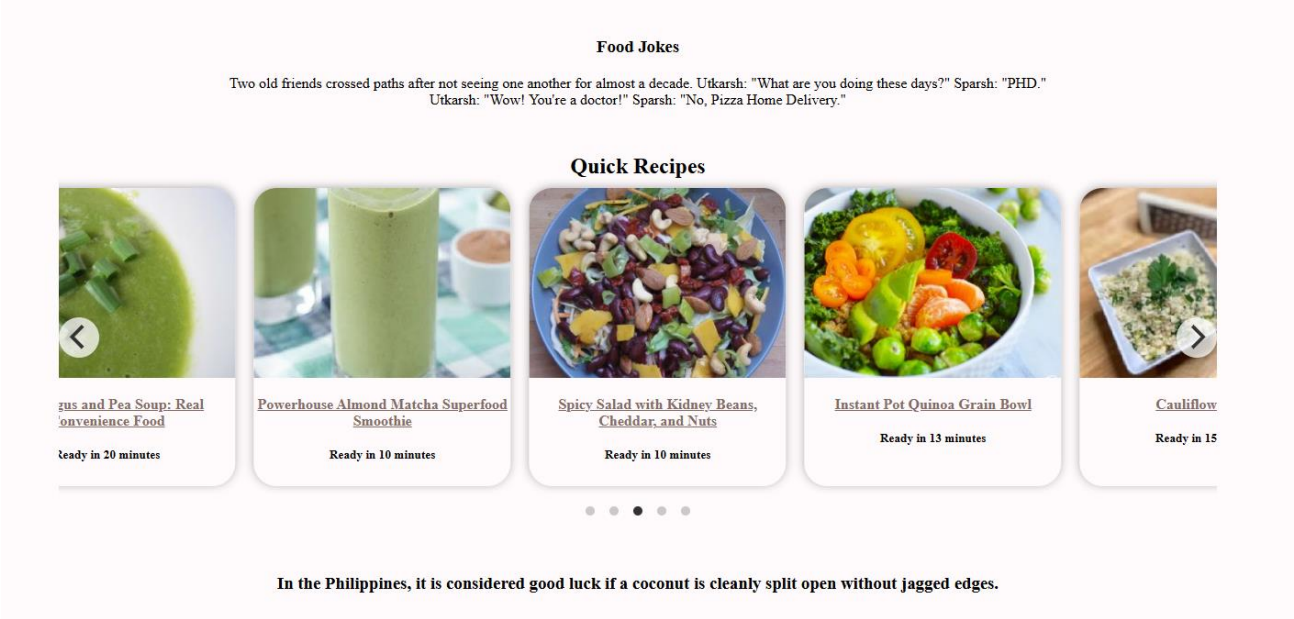

Рисунок 4.17 – Скріншот головної сторінки рецептів з жартами про їжу

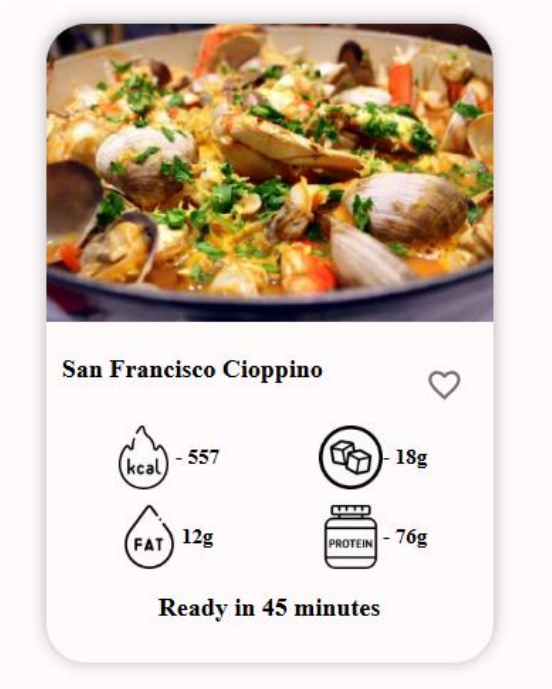

Рисунок 4.18 – Скріншот картки рецепту

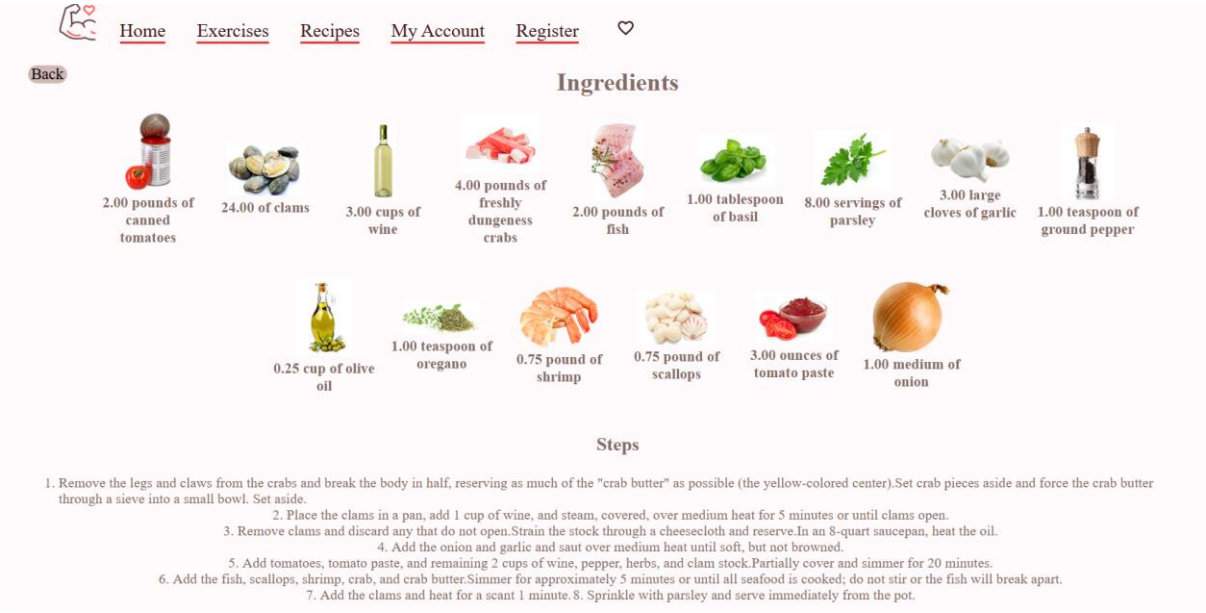

Рисунок 4.19 – Скріншот сторінки детального перегляду рецепту з нутрієнтами, кроками та лайком

Усі улюблені рецепти та вправи відображаються на окремій сторінці "Улюблене". Там вони мають такий самий вигляд, як і на головній сторінці – з повною інформацією та інтерактивними елементами.

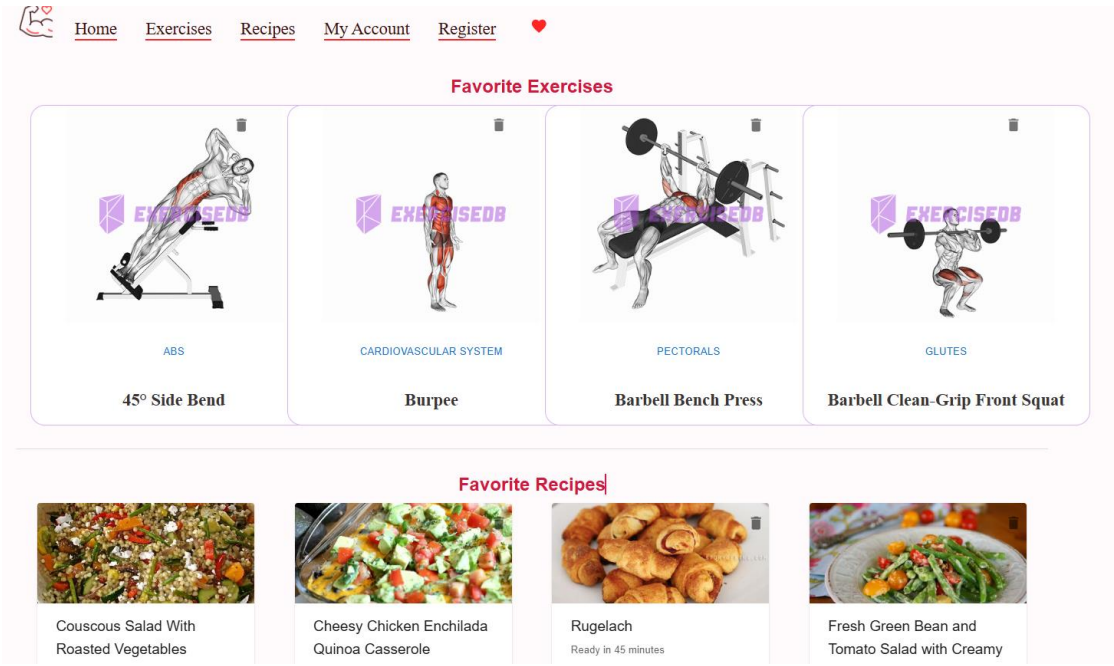

Рисунок 4.20 – Скріншот сторінки улюбленого

Користувач також може вийти з облікового запису натиснувши кнопку "Logout" у правому верхньому куті. Після цього доступ до персональних сторінок закривається, і знову відображаються лише форми логіну та реєстрації.

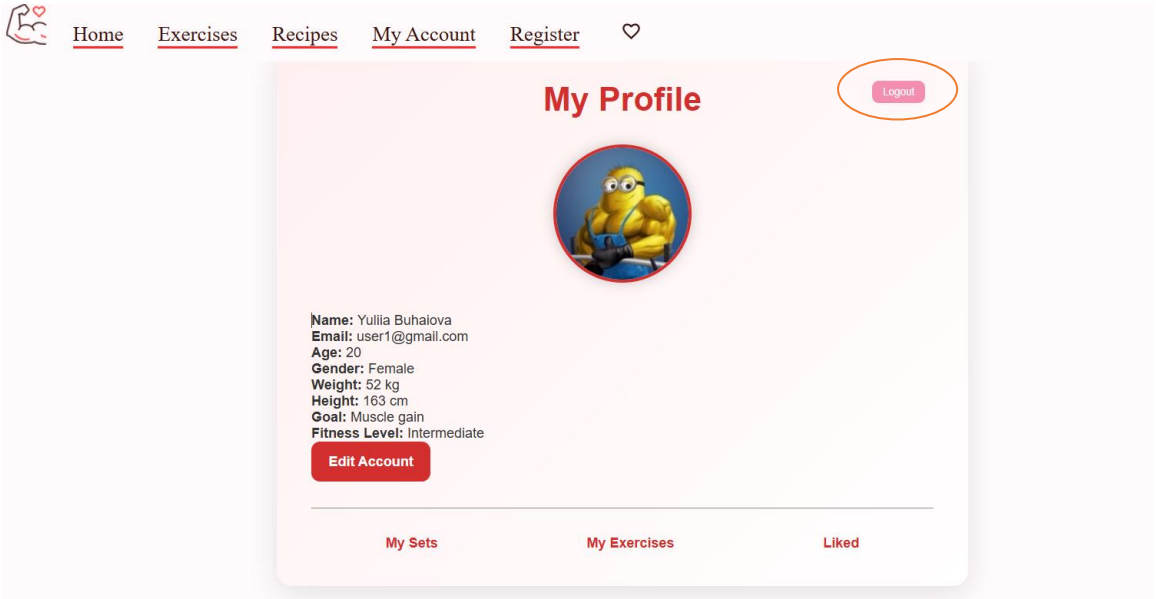

Рисунок 4.21 – Скріншот верхнього меню з кнопкою Logout

Таким чином, застосунок має зрозумілу навігацію, послідовну логіку взаємодії та чітку візуальну структуру, що дозволяє користувачу зручно виконувати всі необхідні дії. Усі ключові функції супроводжуються візуальними елементами, повідомленнями про успішність дій та інтуїтивно зрозумілими кнопками. Вставлені скріншоти ілюструють кожен етап взаємодії користувача з системою.

4.3 Тестування

Метою тестування є перевірка працездатності основного функціоналу інформаційної системи та коректності взаємодії з базою даних. Тестування проводилось вручну відповідно до заздалегідь визначених сценаріїв, які охоплюють як стандартні випадки використання (позитивні тести), так і некоректні дії користувача (негативні тести).

У процесі тестування було перевірено такі основні компоненти:

Реєстрація та логін. При введенні коректного email та надійного паролю користувач успішно потрапляє на головну сторінку. У разі помилки валідації форми або неправильного паролю виводиться відповідне повідомлення.

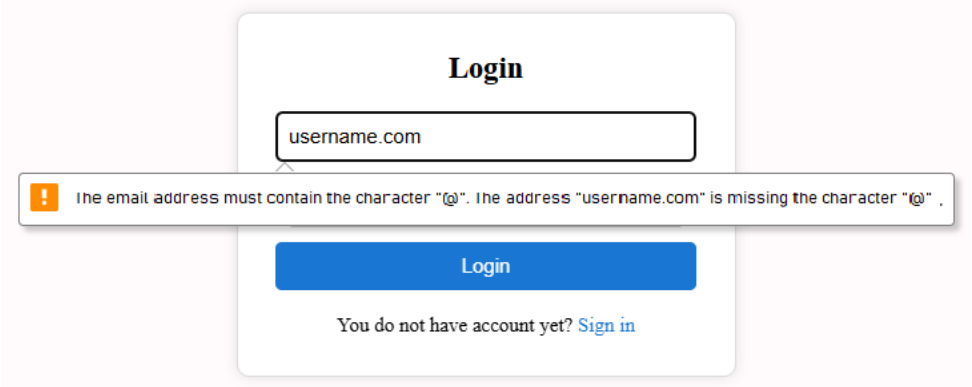

Рисунок 4.22 – Повідомлення про помилку під час входу з некоректною

поштою

Редагування профілю (My Account). Після входу користувач може змінити дані профілю (ім'я, вага, зріст тощо). Збережені дані фіксуються у Firestore і відображаються при повторному відкритті сторінки.

Рисунок 4.23 – Збережена форма профілю зі зміною зросту

Створення власної вправи. Була перевірена форма додавання вправи, зокрема валідація заповнення полів та збереження введеної інформації в базі. Після підтвердження вправа одразу з'являється у списку «Мої вправи».

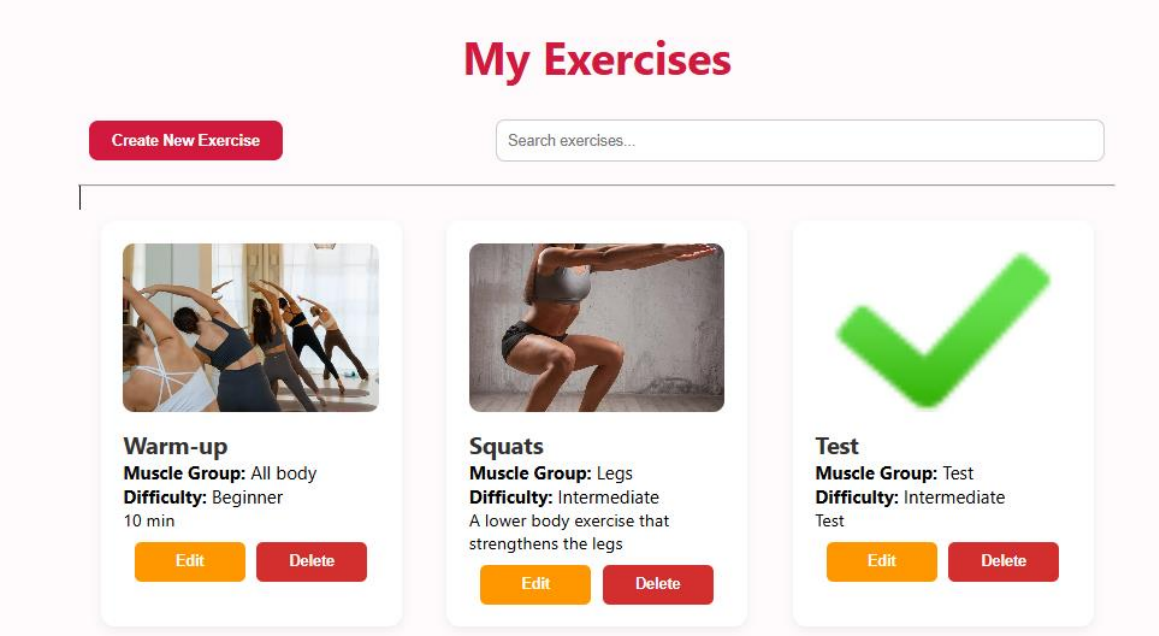

Рисунок 4.24 – Створення нової вправи та її відображення у списку

Створення сету та таймер. Користувач створює набір вправ, зберігає його та запускає. Після запуску відображається таймер, який працює відповідно до логіки перемикання між вправами. Була перевірена коректність таймера, а також можливість зупинки й перезапуску.

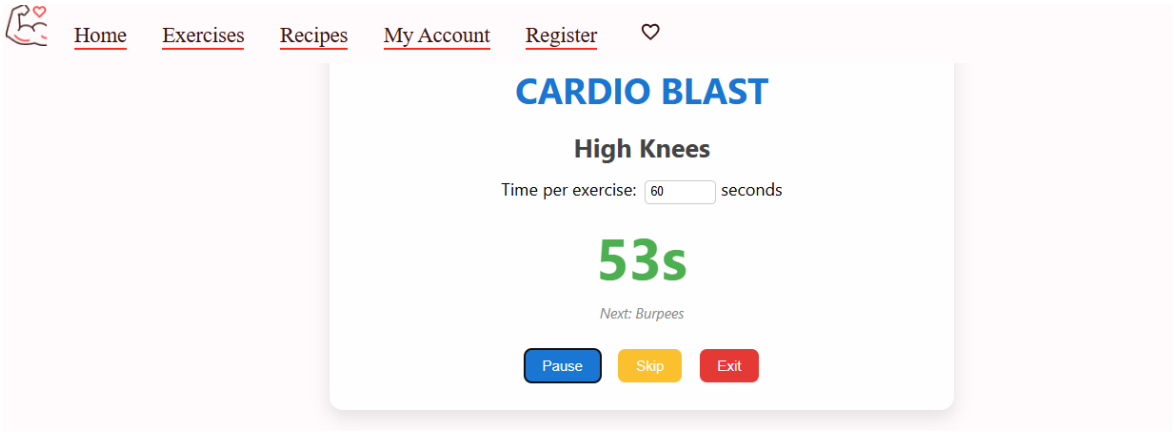

Рисунок 4.25 – Активний таймер під час виконання сету

Додавання до улюбленого. При натисканні іконки «серце» елемент успішно зберігається до списку «Улюблене». Після оновлення сторінки улюблені елементи продовжують відображатись.

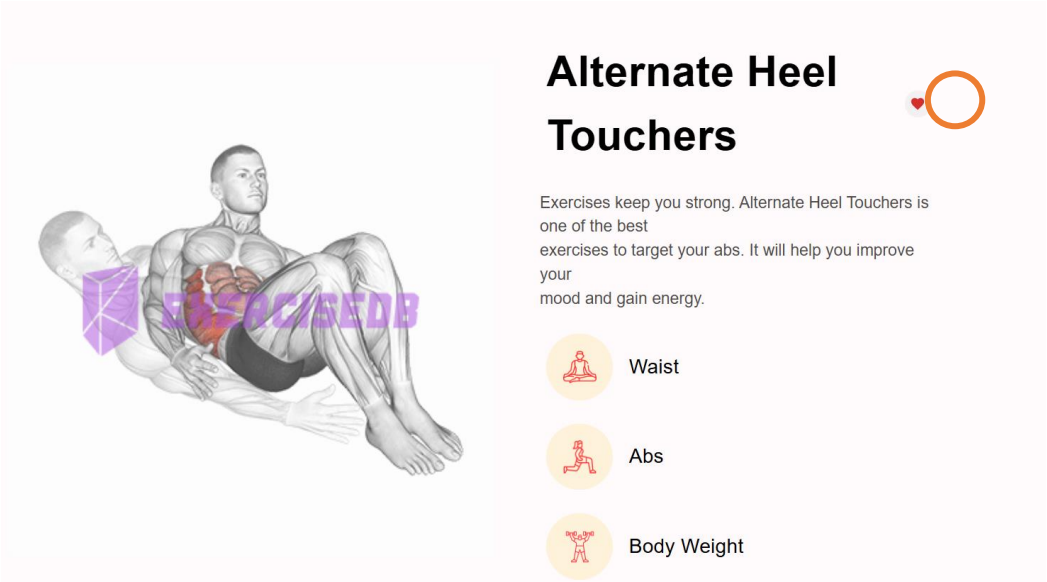

Рисунок 4.26 – Процес додання вправи у улюблене

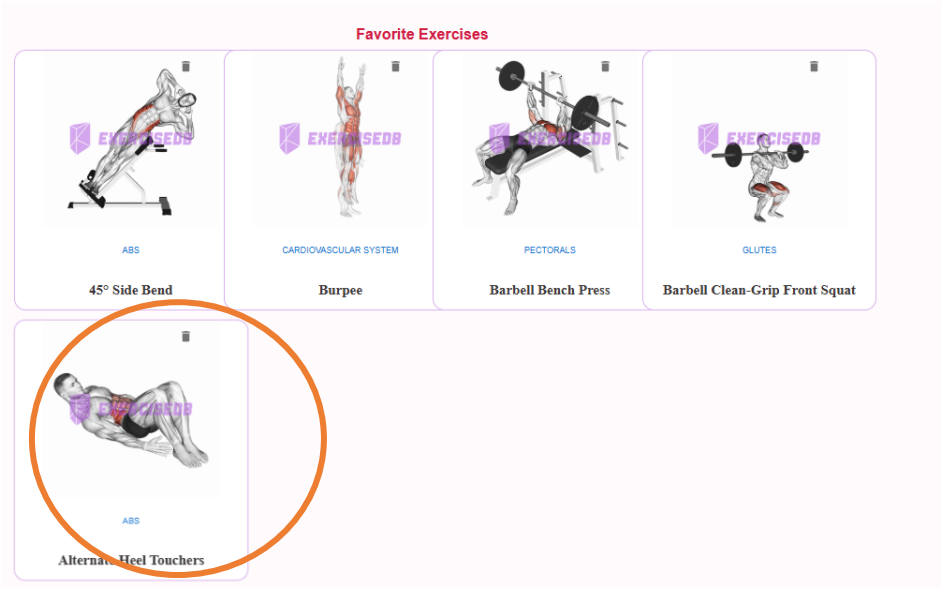

Рисунок 4.27 – Додана вправа в список улюбленого

Пошук вправ. Реалізований пошук було перевірено на випадки введення

валідних та невалідних запитів. При правильному введенні ключового слова виводяться відповідні вправи. У разі порожнього або помилкового запиту результати не відображаються.

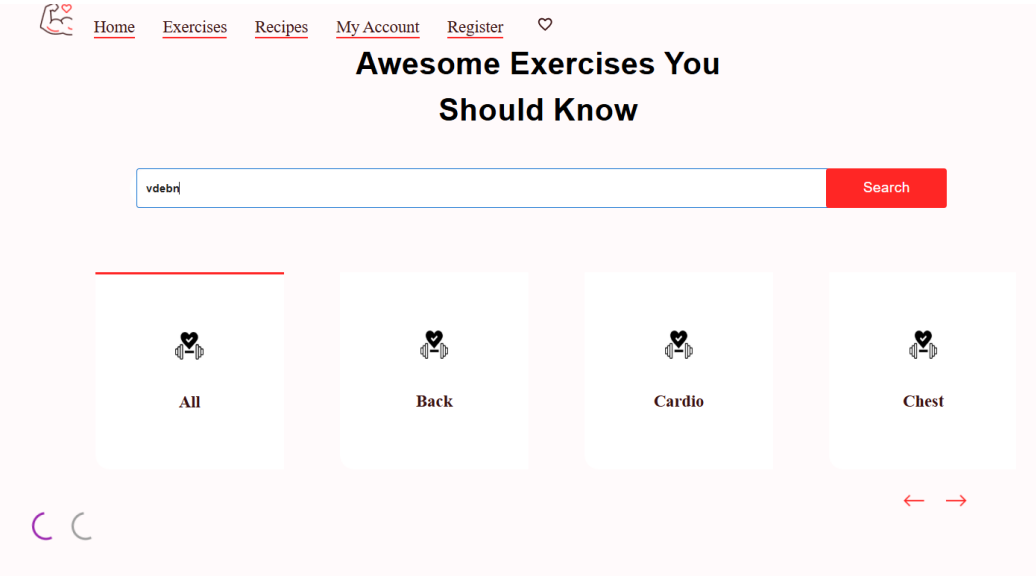

Рисунок 4.28 – Сторінка результатів пошуку з відсутністю результатів

Перегляд рецептів. Перевірено відкриття картки рецепта, відображення опису, інгредієнтів, нутрієнтів, кроків приготування. Усі дані відображаються відповідно до API.

Рисунок 4.29 – Детальний перегляд рецепта з нутрієнтами

Logout. Після натискання кнопки виходу користувач повертається на сторінку логіну. Персональні розділи більше недоступні до повторної авторизації.

Рисунок 4.30 – Повідомлення про вихід із сторінки користувача

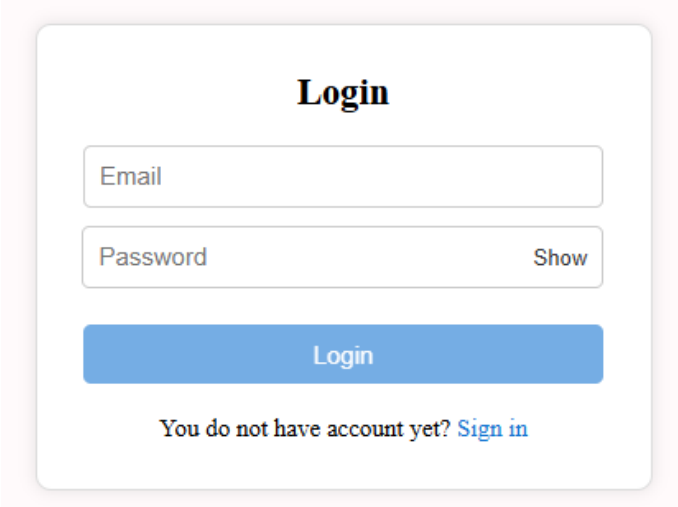

Рисунок 4.31 – Сторінка входу після виходу з аккаунту

Усі основні функції застосунку пройшли тестування. Виявлені помилки були несуттєвими та мали лише візуальний характер. Загалом функціонал працює відповідно до поставлених вимог і забезпечує користувача інтуїтивно зрозумілим та стабільним інтерфейсом для взаємодії з платформою.

Висновки до розділу 4

У даному розділі було детально описано програмну реалізацію інформаційної системи, орієнтованої на фітнес та здорове харчування. Під час розробки системи реалізовано сучасну архітектуру React-компонентів із використанням зовнішніх API (ExerciseDB, YouTube, Spoonacular) та інтеграцією з хмарною платформою Firebase для автентифікації користувачів та зберігання даних у Firestore.

Було створено функціонал реєстрації та авторизації користувачів із валідацією даних і відповідною обробкою помилок. У профілі користувача реалізовано можливість редагування особистих даних (ім'я, дата народження, вага, зріст, email), що дозволяє персоналізувати досвід користування сайтом.

Особливу увагу приділено розділам вправ і рецептів. Користувач має змогу переглядати готові вправи з відеоінструкціями, шукати їх за ключовими словами або частинами тіла, створювати власні вправи, формувати сети й запускати таймер для їх виконання. У розділі рецептів реалізовано автоматичну підбірку випадкових страв, швидких рецептів, а також детальну інформацію про кожен рецепт, включаючи інгредієнти, кроки приготування та харчову цінність.

Функція додавання до улюбленого реалізована як для вправ, так і для рецептів, що забезпечує зручність повторного доступу до контенту. Сторінка "Улюблене" надає повну інформацію про збережені елементи у такому ж форматі, як на основних сторінках.

Підсумовуючи, можна стверджувати, що розроблений застосунок забезпечує повний функціонал для користувачів, зацікавлених у здоровому способі життя. Інтерфейс сайту є інтуїтивно зрозумілим, а реалізовані функції – логічно структурованими, що в цілому забезпечує якісний досвід користування та відповідає поставленим цілям дипломної роботи.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно реалізовано повноцінну інформаційну систему, що слугує цифровим інструментом для підтримки здорового способу життя. На всіх етапах дослідження – від аналізу предметної області до технічної реалізації й тестування – було дотримано логічної послідовності, а отримані результати підтвердили актуальність і доцільність обраного напряму.

Під час аналітичного етапу було обґрунтовано потребу в персоналізованому застосунку, здатному адаптуватися до фізіологічних параметрів та цілей користувача. Було досліджено сучасні тенденції у сфері цифрового здоров’я (digital health), визначено об’єкт, предмет і мету роботи, сформульовано завдання, які лягли в основу подальшої розробки.

На другому етапі було проаналізовано сучасні методи та інформаційні технології, які дозволили реалізувати адаптивну, гнучку й масштабовану архітектуру застосунку. Обрані технології, такі як React, Firebase, зовнішні API, а також принципи UI/UX-дизайну, забезпечили зручність і функціональність системи, що відповідає вимогам сучасного веброзробництва.

У процесі розробки програмного продукту було створено інформаційну систему, яка дозволяє користувачеві створити власний обліковий запис, редагувати дані профілю, зберігати вподобані елементи, створювати персональні вправи й формувати тренувальні сети. Усі дані зберігаються у хмарній базі Firebase Firestore, що гарантує масштабованість, високу швидкодію та гнучке збереження структурованої інформації.

Тестування підтвердило стабільність роботи ключових функцій застосунку, коректну роботу форм, логіку обробки даних, а також відповідність реалізованих функцій заявленим вимогам. Незважаючи на наявність деяких

незавершених елементів, структура та логіка системи дозволяють без перешкод розширити функціонал і запровадити додаткові можливості в майбутньому.

У підсумку, розроблений продукт демонструє комплексний підхід до створення цифрових сервісів у сфері здоров’я. Він здатен слугувати основою для запуску повноцінного сервісу з персоналізованими порадами, навчальним та мотиваційним контентом для користувачів, які прагнуть підтримувати здоровий спосіб життя за допомогою сучасних технологій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. MyFitnessPal: Calorie Counter [Електронний ресурс]. URL: https://apps.apple.com/us/app/myfitnesspal-calorie-counter/id341232718 (дата звернення: 01.05.2025).

2. Fitbit: Health & Fitness [Електронний ресурс]. URL: https://apps.apple.com/us/app/fitbit-health-fitness/id462638897 (дата звернення: 01.05.2025).

3. Strava: Run, Bike, Hike [Електронний ресурс]. URL: https://apps.apple.com/us/app/strava-run-bike-hike/id426826309 (дата звернення: 01.05.2025).

4. Lose It! – Calorie Counter [Електронний ресурс]. URL: https://apps.apple.com/us/app/lose-it-calorie-counter/id297368629 (дата звернення: 01.05.2025).

5. JEFIT Workout Planner Gym Log [Електронний ресурс]. URL: https://apps.apple.com/us/app/jefit-workout-planner-gym-log/id449810000 (дата звернення: 01.05.2025).

6. YAZIO Calorie Counter & Diet [Електронний ресурс]. URL: https://apps.apple.com/ua/app/yazio-calorie-counter-diet/id946099227 (дата звернення: 01.05.2025).

7. 8fit Workouts & Meal Planner [Електронний ресурс]. URL: https://apps.apple.com/us/app/8fit-workouts-meal-planner/id866617777 (дата звернення: 01.05.2025).

8. Lifesum. Healthy Eating & Diet [Електронний ресурс]. URL: https://apps.apple.com/us/app/lifesum-food-calorie-tracker/id286906691 (дата звернення: 01.05.2025).

9. Nike Training Club: Wellness [Електронний ресурс]. URL: https://apps.apple.com/us/app/nike-training-club-wellness/id301521403 (дата звернення: 01.05.2025).

10. Американський коледж спортивної медицини (ACSM). AI-платформи для персоналізованих фітнес-програм [Електронний ресурс]. URL: https://acsm.org/personalized-fitness-mobile-technology/ (дата звернення: 01.05.2025).

11. Digital Trends. Як технології змінюють фітнес: від AI до розумних тренажерів [Електронний ресурс]. URL: https://www.digitaltrends.com/health-fitness/ (дата звернення: 01.05.2025).

12. JavaScript documentation. Mozilla Developer Network. [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 02.05.2025).

13. React – A JavaScript library for building user interfaces. [Електронний ресурс]. URL: https://reactjs.org/docs/getting-started.html (дата звернення: 02.05.2025).

14. HTML5 guide. Mozilla Developer Network. [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5 (дата звернення: 02.05.2025).

15. CSS: Cascading Style Sheets. Mozilla Developer Network. [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 02.05.2025).

16. Sass: Syntactically Awesome Stylesheets. [Електронний ресурс]. URL: https://sass-lang.com/guide (дата звернення: 02.05.2025).

17. Material UI: React components for faster web development. [Електронний ресурс]. URL: https://mui.com/material-ui/getting-started/overview/ (дата звернення: 02.05.2025).

18. React Router – Declarative routing for React. [Електронний ресурс]. URL: https://reactrouter.com/en/main/start/tutorial (дата звернення: 02.05.2025).

19. Flickity: Touch, responsive, flickable carousels. [Електронний ресурс]. URL: https://flickity.metafizzy.co/ (дата звернення: 02.05.2025).

20. ExerciseDB API. RapidAPI. [Електронний ресурс]. URL: https://rapidapi.com/justin-WFnsXH_t6/api/exercisedb (дата звернення: 02.05.2025).

21. Spoonacular Food and Recipe API. [Електронний ресурс]. URL: https://spoonacular.com/food-api (дата звернення: 02.05.2025).

22. YouTube Data API v3. Google Developers. [Електронний ресурс]. URL: https://developers.google.com/youtube/v3/docs/search/list?hl=us (дата звернення: 02.05.2025).

23. Fetch API. Mozilla Developer Network. [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (дата звернення: 02.05.2025).

24. Git documentation. [Електронний ресурс]. URL: https://git-scm.com/doc (дата звернення: 02.05.2025).

25. GitHub Docs. [Електронний ресурс]. URL: https://docs.github.com/en (дата звернення: 02.05.2025).

26. GitHub Pages. [Електронний ресурс]. URL: https://pages.github.com/ (дата звернення: 02.05.2025).

27. Node.js documentation. [Електронний ресурс]. URL: https://nodejs.org/en/docs (дата звернення: 02.05.2025).

28. npm Docs. [Електронний ресурс]. URL: https://docs.npmjs.com/ (дата звернення: 02.05.2025).

29. ReactDOM. React documentation. [Електронний ресурс]. URL: https://reactjs.org/docs/react-dom.html (дата звернення: 02.05.2025).

30. Introducing JSX. React documentation. [Електронний ресурс]. URL: https://reactjs.org/docs/introducing-jsx.html (дата звернення: 02.05.2025).

31. Visual Studio Code [Електронний ресурс]. URL: https://code.visualstudio.com/ (дата звернення: 02.05.2025).

32. Figma [Електронний ресурс]. URL: https://www.figma.com/ (дата звернення: 10.05.2025).

33. Firebase Firestore – NoSQL Database [Електронний ресурс]. URL: https://firebase.google.com/docs/firestore (дата звернення: 09.05.2025).

34. Spoonacular API [Електронний ресурс]. URL: https://spoonacular.com/food-api (дата звернення: 09.05.2025).

35. Programmers Force – The Leading AI Company in UK [Електронний ресурс]. URL: https://programmersforce.co.uk/blogs/why-should-you-use-sass-over-conventional-css/ (дата звернення: 02.05.2025).

36. Zapier – How to use an API: A tutorial for beginners [Електронний ресурс]. URL: https://zapier.com/blog/how-to-use-api/ (дата звернення: 02.05.2025).

ДОДАТОК А

Лістинг фрагментів коду

Register.js

```
import React, { useState } from "react";
import { auth } from "../../firebase";
import { createUserWithEmailAndPassword } from "firebase/auth";
import { Link, useNavigate } from "react-router-dom";

const Register = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const isPasswordValid = (pwd) => {
    const hasUppercase = /[A-Z]/.test(pwd);
    const hasNumber = /[0-9]/.test(pwd);
    const isLongEnough = pwd.length >= 6;
    return hasUppercase && hasNumber && isLongEnough;
  };

  const handleRegister = async (e) => {
    e.preventDefault();
    setError("");

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailRegex.test(email)) {
      setError("Please enter a valid email address.");
      return;
    }

    if (!isPasswordValid(password)) {
      setError("Password must be at least 6 characters long, include an
uppercase letter and a number.");
      return;
    }

    try {
      await createUserWithEmailAndPassword(auth, email, password);
      alert("Registration successful!");
      navigate("/login");
    } catch (err) {
      setError(err.message);
    }
  };
```

Login.js

```
import React, { useState } from "react";
import { auth } from "../../firebase";
import { signInWithEmailAndPassword, signOut } from "firebase/auth";
import { Link, useNavigate } from "react-router-dom";
```

```
      const Login = () => {
        const [email, setEmail] = useState("");
        const [password, setPassword] = useState("");
        const [showPassword, setShowPassword] = useState(false);
        const [error, setError] = useState("");
        const [success, setSuccess] = useState("");
        const navigate = useNavigate();

        const isPasswordValid = (pwd) => {
          const hasUppercase = /[A-Z]/.test(pwd);
          const hasNumber = /[0-9]/.test(pwd);
          const isLongEnough = pwd.length >= 6;
          return hasUppercase && hasNumber && isLongEnough;
        };

        const handleLogin = async (e) => {
          e.preventDefault();
          setError("");
          setSuccess("");

          const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
          if (!emailRegex.test(email)) {
            setError("Please enter a valid email address.");
            return;
          }

          if (!isPasswordValid(password)) {
            setError("Password must be at least 6 characters long, include an
uppercase letter and a number.");
            return;
          }

          try {
            await signInWithEmailAndPassword(auth, email, password);
            setSuccess("Login successful!");
            setEmail("");
            setPassword("");
            navigate("/");
          } catch (err) {
            setError(err.message);
          }
        };

        const handleLogout = async () => {
          try {
            await signOut(auth);
            navigate("/login");
          } catch (err) {
            setError(err.message);
          }
        };

        const inputStyle = {
          display: "block",
          marginBottom: "12px",
          width: "100%",
```

```
    padding: "10px",
    borderRadius: "5px",
    border: "1px solid #ccc",
    fontSize: "16px",
  };
```

App.js

```
import React from 'react';
import { Route, Routes } from 'react-router-dom';
import { Box } from '@mui/material';

import './App.css';
import ExerciseDetail from './pages/ExerciseDetails/ExerciseDetail';
import Home from './pages/Home/Home';
import Navbar from './components/Navbar/Navbar';
import Footer from './components/Footer/Footer';
import { Main_recipes } from "./pages/Recipes/mainRecipes/main_recipes";
import { ExercisesMain } from "./pages/Exercises/Exercises";
import { RecipeSide } from "./pages/Recipes/recipeCard/RecipeSide/recipeSide";
import MyExercises from './pages/CustomExercises/MyExercises';
import MySets from './pages/MySets/MySets';
import TimerPage from './pages/SetTimer/TimerPage';


import Login from "./pages/Auth/Login";
import Register from "./pages/Auth/Register";
import Liked from './pages/Liked/Liked';

import MyAccount from "./pages/MyAccount/MyAccount";
import { LikedProvider } from './context/LikedContext';
const App = () => (
  <LikedProvider>
    <Box width="400px" sx={{ width: { xl: '1488px' } }} m="auto">
      <Navbar />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/exercises" element={<ExercisesMain />} />
        <Route path="/exercise/:id" element={<ExerciseDetail />} />
        <Route path="/recepies/:id" element={<RecipeSide />} />
        <Route path="/recipes" element={<Main_recipes />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />
        <Route path="/my-account" element={<MyAccount />} />
        <Route path="/liked" element={<Liked />} />
        <Route path="/my-exercises" element={<MyExercises />} />
        <Route path="/my-sets" element={<MySets/>} />
        <Route path="/sets/:setId" element={<TimerPage />} />

      </Routes>
      <Footer />
    </Box>
  </LikedProvider>
);
```

```
export default App;
```

Firebase.js

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import {
getFirestore,
collection,
addDoc,
getDocs,
updateDoc,
deleteDoc,
doc,
setDoc,
getDoc,
query,
where
} from "firebase/firestore";

// Firebase конфіг
const firebaseConfig = {
apiKey: "AIzaSyBpCiWVZTvqYxrcgX7frevI9z7WM9rkfxo",
authDomain: "fitnessapp-e09eb.firebaseapp.com",
projectId: "fitnessapp-e09eb",
storageBucket: "fitnessapp-e09eb.appspot.com",
messagingSenderId: "218069016353",
appId: "1:218069016353\:web:28d5d43a0d72e740502860"
};

const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
auth.languageCode = 'en';
export const db = getFirestore(app);
export const createExercise = async (exerciseData) => {
return await addDoc(collection(db, "customExercises"), exerciseData);
};

export const getExercises = async (userId) => {
const q = query(collection(db, "customExercises"), where("userId", "==",
userId));
const querySnapshot = await getDocs(q);
return querySnapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
};

export const updateExercise = async (id, updatedData) => {
const docRef = doc(db, "customExercises", id);
return await updateDoc(docRef, updatedData);
};

export const deleteExercise = async (id) => {
return await deleteDoc(doc(db, "customExercises", id));
};

export const createSet = async (setData) => {
return await addDoc(collection(db, "sets"), setData);
```

```
};

export const getSets = async (userId) => {
const q = query(collection(db, "sets"), where("userId", "==", userId));
const querySnapshot = await getDocs(q);
return querySnapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
};

export const addToFavorites = async (userId, itemId, type) => {
const docRef = doc(db, "favorites", `${userId}_${itemId}`);
return await setDoc(docRef, { userId, itemId, type });
};

export const getFavorites = async (userId, type) => {
const q = query(collection(db, "favorites"), where("userId", "==", userId),
where("type", "==", type));
const querySnapshot = await getDocs(q);
return querySnapshot.docs.map(doc => doc.data().itemId);
};

export const saveUserProfile = async (userId, profileData) => {
const docRef = doc(db, "users", userId);
return await setDoc(docRef, profileData);
};

export const getUserProfile = async (userId) => {
const docRef = doc(db, "users", userId);
const docSnap = await getDoc(docRef);
return docSnap.exists() ? docSnap.data() : null;
};
```

recipeSide.js

```
import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import styles from './recipeSide.scss'
import {Ingredients} from "./Ingredients/Ingredients";
import {ItemCard} from "./Item/item";
import {Steps} from "./Steps/Steps";


export const RecipeSide = () => {
  const [recipeInfo, setRecipeInfo] = useState();
  const [recipeCalories, setRecipeCalories] = useState();
  const { id } = useParams();
  let baseURL = 'https://api.spoonacular.com'
  let apiKey= '73f3eac60a9f447d8d7e39438747afd3'
  useEffect(() => {
    const getRecipeData = async()=>
    {
      const                response                =                await
fetch(`https://api.spoonacular.com/recipes/${id}/information?addRecipeInformatio
n=true&apiKey=${apiKey}`);
      const data = await response.json();

      setRecipeInfo(data)
    };
```

```
          const getCaloriesData = async()=>
          {

            const                response                =                await
fetch(`${baseURL}/recipes/${id}/nutritionWidget.json?apiKey=${apiKey}`);
            const data = await response.json();

            setRecipeCalories(data)
          };

            getCaloriesData(), getRecipeData()
        }, []);

        if (!recipeCalories||!recipeInfo) return <div>No Data</div>;
```

main_recipes.js

```
      import React, { useEffect, useState } from "react";
      import { useParams } from "react-router-dom";
      import { TextField } from "@mui/material";
      import { RecommendRecipe } from "../recommendRecipes/recommendRecipe";
      import Loader from "../../../components/Loader/Loader";
      import { QuickRecipes } from "../quickRecipes/quickRecipes";
      import HorizontalScroll, { Scroll, ScrollBar } from "../scroll/scrlbar";
      import s from './main_recipes.scss';

      export const Main_recipes = () => {
        const [trivia, setTrivia] = useState('');
        const [quickRecipes, setQuickRecipes] = useState({});
        const [mainRecipes, setMainRecipes] = useState([]);
        const [jokes, setJokes] = useState('');
        const [search, setSearch] = useState('');
        const [query, setQuery] = useState('');
        const [find, setFind] = useState('');
        const { id } = useParams();

        const baseURL = 'https://api.spoonacular.com';
        const apiKey = '3eafa5a00c4b43859eb81177e2e47cef';

        const searchData = async () => {
          const response = await fetch(

`${baseURL}/recipes/complexSearch?query=${query}&number=5&offset=0&addRecipeInfo
rmation=true&apiKey=${apiKey}`
          );
          const data = await response.json();
          setSearch(data);
          console.log(data);
        };

        const handleChange = (event) => {
          setQuery(event.target.value);
        };

        useEffect(() => {
          const fetchData = async () => {
```

```
            try {
              const [recipesRes, triviaRes, jokesRes, quickRes] = await
Promise.all([
                fetch(`${baseURL}/recipes/random?number=10&apiKey=${apiKey}`),
                fetch(`${baseURL}/food/trivia/random?apiKey=${apiKey}`),
                fetch(`${baseURL}/food/jokes/random?apiKey=${apiKey}`),

fetch(`${baseURL}/recipes/complexSearch?number=5&addRecipeInformation=true&maxRe
adyTime=20&apiKey=${apiKey}`)
              ]);

              const recipesData = await recipesRes.json();
              const triviaData = await triviaRes.json();
              const jokesData = await jokesRes.json();
              const quickData = await quickRes.json();

              setMainRecipes(recipesData.recipes || []);
              setTrivia(triviaData.text || '');
              setJokes(jokesData.text || '');
              setQuickRecipes(quickData || {});
            } catch (error) {
              console.error("Fetch error:", error);
            }
          };

          fetchData();
        }, []);

        if (!mainRecipes.length || !trivia || !jokes || !quickRecipes) return
<Loader />;

        return (
          <div className={'wrap'} style={s}>
            <div className={'main'} style={s}>
              <div className={'text'}>
                <div className={'h2'}>Recipes of the day</div>
                <Scroll props={mainRecipes.slice(0, 5)} />
              </div>

              <div className={'text'}>
                <div className={'h2'}>Recommend for you</div>
                <Scroll props={mainRecipes.slice(5, 11)} />
              </div>

              <div
                style={{
                  marginBottom: '50px',
                  display: 'flex',
                  justifyContent: 'center',
                  alignItems: 'center',
                  flexDirection: 'column'
                }}
              >
                <h3 className={'text'} style={{ width: '70%', textAlign: 'center',
marginBottom: '20px' }}>
                  Food Jokes
```

```
              </h3>
              <p style={{ textAlign: 'center', width: '70%' }}>{jokes}</p>
            </div>

            <div className={'text'}>
              <div className={'h2'}>Quick Recipes</div>
              <QuickRecipes props={quickRecipes} />
            </div>

            <div style={{ marginTop: '50px', textAlign: 'center' }}>
              <h3 className={'text'}>{trivia}</h3>
            </div>
          </div>
        </div>
      );
    };
```

ExerciseDetail.js

```
import React, { useEffect, useState } from 'react';
    import { useParams } from 'react-router-dom';
    import { Box } from '@mui/material';

    import { exerciseOptions, fetchData, youtubeOptions } from
'../../utils/fetchData';
    import Detail from '../../components/Details/Detail';
    import ExerciseVideos from '../../components/Exercises/ExerciseVideos';
    import SimilarExercises from '../../components/Exercises/SimilarExercises';



    const ExerciseDetail = () => {
      const [exerciseDetail, setExerciseDetail] = useState({});
      const [exerciseVideos, setExerciseVideos] = useState([]);
      const [targetMuscleExercises, setTargetMuscleExercises] = useState([]);
      const [equipmentExercises, setEquipmentExercises] = useState([]);
      const { id } = useParams();

      useEffect(() => {
        window.scrollTo({ top: 0, behavior: 'smooth' });

        const fetchExercisesData = async () => {
          const exerciseDbUrl = 'https://exercisedb.p.rapidapi.com';
          const youtubeSearchUrl = 'https://youtube-search-and-
download.p.rapidapi.com';
          const exerciseDetailData = await
fetchData(`${exerciseDbUrl}/exercises/exercise/${id}`, exerciseOptions);
          setExerciseDetail(exerciseDetailData);

          const exerciseVideosData = await
fetchData(`${youtubeSearchUrl}/search?query=${exerciseDetailData.name}
exercise`, youtubeOptions);
          setExerciseVideos(exerciseVideosData.contents);

          const targetMuscleExercisesData = await
fetchData(`${exerciseDbUrl}/exercises/target/${exerciseDetailData.target}`,
```

```
exerciseOptions);
            setTargetMuscleExercises(targetMuscleExercisesData);

            const            equimentExercisesData            =            await
fetchData(`${exerciseDbUrl}/exercises/equipment/${exerciseDetailData.equipment}`
, exerciseOptions);
            setEquipmentExercises(equimentExercisesData);
          };

           fetchExercisesData();
        }, [id]);

        if (!exerciseDetail) return <div>No Data</div>;

        return (
          <Box sx={{ mt: { lg: '96px', xs: '60px' } }}>
            <Detail exerciseDetail={exerciseDetail} />
            <ExerciseVideos                        exerciseVideos={exerciseVideos}
name={exerciseDetail.name} />
            <SimilarExercises       targetMuscleExercises={targetMuscleExercises}
equipmentExercises={equipmentExercises} />
          </Box>
        );
      };

      export default ExerciseDetail;
```

ExerciseVideos.js

```
      import React from 'react';
      import { Typography, Box, Stack } from '@mui/material';
      import Loader from '../Loader/Loader';


      const ExerciseVideos = ({ exerciseVideos, name }) => {
        if (!exerciseVideos.length) return <Loader/>;

        return (
          <Box sx={{ marginTop: { lg: '203px', xs: '20px' } }} p="20px">
            <Typography  sx={{  fontSize:  {  lg:  '44px',  xs:  '25px'  }  }}
fontWeight={700} color="#000" mb="33px">
              Watch <span style={{ color: '#FF2625', textTransform: 'capitalize'
}}>{name}</span> exercise videos
            </Typography>
            <Stack sx={{ flexDirection: { lg: 'row' }, gap: { lg: '110px', xs:
'0px' } }} justifyContent="flex-start" flexWrap="wrap" alignItems="center">
              {exerciseVideos?.slice(0, 3)?.map((item, index) => (
                <a
                  key={index}
                  className="exercise-video"
                  href={`https://www.youtube.com/watch?v=${item.video.videoId}`}
                  target="_blank"
                  rel="noreferrer"
                >
                  <img      style={{      borderTopLeftRadius:      '20px'      }}
src={item.video.thumbnails[0].url} alt={item.video.title} />
```

```
                  <Box>
                    <Typography sx={{ fontSize: { lg: '28px', xs: '18px' } }}
fontWeight={600} color="#000">
                      {item.video.title}
                    </Typography>
                    <Typography fontSize="14px" color="#000">
                      {item.video.channelName}
                    </Typography>
                  </Box>
                </a>
              ))}
            </Stack>
          </Box>
        );
      };

      export default ExerciseVideos;
```

LikedContext.js

```
      import React, { createContext, useContext, useEffect, useState } from
'react';
      import { auth, db } from '../firebase';
      import {
        collection,
        query,
        where,
        getDocs,
        setDoc,
        deleteDoc,
        doc
      } from 'firebase/firestore';

      const LikedContext = createContext();

      export const LikedProvider = ({ children }) => {
        const [likedItems, setLikedItems] = useState([]);
        const user = auth.currentUser;

        useEffect(() => {
          const fetchLiked = async () => {
            if (user) {
              const q = query(collection(db, 'favorites'), where('userId', '==',
user.uid));
              const querySnapshot = await getDocs(q);
              const items = querySnapshot.docs.map(doc => ({
                id: doc.data().itemId,
                type: doc.data().type,
              }));
              setLikedItems(items);
            }
          };
          fetchLiked();
        }, [user]);

        const toggleLike = async (item, type) => {
```

```
    if (!user) return;

    const docId = `${user.uid}_${item.id}`;
    const alreadyLiked = likedItems.some(i => i.id === item.id);

    if (alreadyLiked) {
      await deleteDoc(doc(db, 'favorites', docId));
      setLikedItems(prev => prev.filter(i => i.id !== item.id));
    } else {
      await setDoc(doc(db, 'favorites', docId), {
        userId: user.uid,
        itemId: item.id,
        type: type,
      });
      setLikedItems(prev => [...prev, { id: item.id, type }]);
    }
  };

  const isLiked = (id) => likedItems.some(i => i.id === id);

  return (
    <LikedContext.Provider value={{ likedItems, toggleLike, isLiked }}>
      {children}
    </LikedContext.Provider>
  );
};

export const useLike = () => useContext(LikedContext);
```

MyAccount.jsx

```
import React, { useState, useEffect } from "react";
import { Link, useNavigate } from "react-router-dom";
import { doc, getDoc, setDoc } from "firebase/firestore";
import { auth, db } from "../../firebase";

const MyAccount = () => {
  const navigate = useNavigate();
  const user = auth.currentUser;
  const userId = user?.uid || "demoUser";

  const [userData, setUserData] = useState({
    displayName: "",
    email: "",
    avatarUrl: "",
    age: "",
    gender: "",
    weight: "",
    height: "",
    goal: "",
    level: ""
  });

  const [editing, setEditing] = useState(false);
  const [tempPassword, setTempPassword] = useState("");
  const [error, setError] = useState("");
```

```
  useEffect(() => {
    const fetchUserData = async () => {
      const docRef = doc(db, "users", userId);
      const docSnap = await getDoc(docRef);
      if (docSnap.exists()) {
        setUserData(docSnap.data());
      }
    };
    fetchUserData();
  }, [userId]);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setUserData((prev) => ({ ...prev, [name]: value }));
  };

  const handleAvatarChange = (e) => {
    const file = e.target.files[0];
    if (file) {
      const reader = new FileReader();
      reader.onloadend = () => {
        setUserData((prev) => ({ ...prev, avatarUrl: reader.result }));
      };
      reader.readAsDataURL(file);
    }
  };

  const handleSave = async () => {
    setError("");
    if (!userData.displayName.trim()) {
      setError("Display Name cannot be empty");
      return;
    }
    if (!userData.email.includes("@")) {
      setError("Invalid email");
      return;
    }
    try {
      await setDoc(doc(db, "users", userId), userData);
      setEditing(false);
    } catch (err) {
      setError("Failed to save data");
    }
  };

  const handleLogout = () => {
    auth.signOut();
    navigate("/login");
  };
};

export default MyAccount;
```

MyExercises.jsx

```
import React, { useState, useEffect } from "react";
import { db, auth } from "../../firebase";
import {
  collection,
  addDoc,
  onSnapshot,
  deleteDoc,
  doc,
  updateDoc,
  query,
  where
} from "firebase/firestore";

const MyExercises = () => {
  const user = auth.currentUser;
  const [exercise, setExercise] = useState({
    name: "",
    description: "",
    muscleGroup: "",
    difficulty: "",
    imageUrl: "",
  });

  const [exercises, setExercises] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");
  const [showForm, setShowForm] = useState(false);
  const [editId, setEditId] = useState(null);

  useEffect(() => {
    if (!user) return;
    const q = query(collection(db, "customExercises"), where("userId", "==",
user.uid));
    const unsubscribe = onSnapshot(q, (snapshot) => {
      const data = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data()
}));
      setExercises(data);
    });
    return unsubscribe;
  }, [user]);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setExercise((prev) => ({ ...prev, [name]: value }));
  };

  const handleImageChange = (e) => {
    const file = e.target.files[0];
    if (file) {
      const reader = new FileReader();
      reader.onloadend = () => {
        setExercise((prev) => ({ ...prev, imageUrl: reader.result }));
      };
      reader.readAsDataURL(file);
    }
  };
```

```
        const handleSave = async () => {
          if (!exercise.name || !exercise.description || !exercise.muscleGroup ||
!exercise.difficulty) return;
          const data = { ...exercise, userId: user.uid };

          if (editId) {
            await updateDoc(doc(db, "customExercises", editId), data);
          } else {
            await addDoc(collection(db, "customExercises"), data);
          }

          setExercise({ name: "", description: "", muscleGroup: "", difficulty:
"", imageUrl: "" });
          setEditId(null);
          setShowForm(false);
        };

        const handleDelete = async (id) => {
          await deleteDoc(doc(db, "customExercises", id));
        };

        const handleEdit = (ex) => {
          setExercise({ ...ex });
          setEditId(ex.id);
          setShowForm(true);
        };

        const filteredExercises = exercises.filter((ex) =>
          ex.name.toLowerCase().includes(searchTerm.toLowerCase())
        );
```

MySets.jsx

```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { db } from "../../firebase";
import {
  collection,
  addDoc,
  getDocs,
  deleteDoc,
  updateDoc,
  doc,
} from "firebase/firestore";

const MySets = () => {
  const navigate = useNavigate();
  const [sets, setSets] = useState([]);
  const [newSet, setNewSet] = useState({
    name: "",
    exercises: [],
    totalTime: "",
    difficulty: ""
  });
  const [showForm, setShowForm] = useState(false);
  const [searchTerm, setSearchTerm] = useState("");
```

```
const setsRef = collection(db, "sets");

useEffect(() => {
  const fetchSets = async () => {
    const data = await getDocs(setsRef);
    setSets(data.docs.map((doc) => ({ ...doc.data(), id: doc.id })));
  };
  fetchSets();
}, []);

const handleChange = (e) => {
  const { name, value } = e.target;
  setNewSet((prev) => ({ ...prev, [name]: value }));
};

const handleAddSet = async (e) => {
  e.preventDefault();
  if (!newSet.name || !newSet.totalTime || !newSet.difficulty) return;
  const docRef = await addDoc(setsRef, newSet);
  setSets([...sets, { ...newSet, id: docRef.id }]);
  setNewSet({ name: "", exercises: [], totalTime: "", difficulty: "" });
  setShowForm(false);
};

const handleDeleteSet = async (id) => {
  const setDoc = doc(db, "sets", id);
  await deleteDoc(setDoc);
  setSets(sets.filter((s) => s.id !== id));
};

const handleEditSet = async (setToEdit) => {
  setNewSet(setToEdit);
  setShowForm(true);
};
```