СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) Інформаційних технологій та електроніки

Кафедра Інформаційних технологій та програмування
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
(бакалавр, магістр)
спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
спеціалізація Інженерія програмного забезпечення
(назва спеціалізації)

на тему Мобільний застосунок для відстеження та аналізу фізичної активності

Виконав: студент групи ІПЗ-20бз _____ О.В. Мірошник
(підпис) (ініціали і прізвище)

Керівник _____ В.Г. Іванов
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ О. І. Захожай
(підпис) (ініціали і прізвище)

Рецензент Доц., д.т.н. Лифар В.О.

Київ – 2024

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) Інформаційних технологій та електроніки.
Кафедра Інформаційних технологій та програмування
(повна назва кафедри)
Освітній ступінь бакалавр
(бакалавр, магістр)
спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
спеціалізація Інженерія програмного забезпечення
(назва спеціалізації)

_____ Захожай О.І.
"___" _____ 2024 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Мірошник Олександр Васильович
(прізвище, ім'я, по-баткові)

1. Тема роботи Мобільний застосунок для відстеження та аналізу фізичної активності_____
_____
_____

Керівник роботи Іванов Віталій Геннадійович, Доц., к.т.н._____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від "___"_____20__року №___

2.Строк подання студентом роботи_____
3. Вихідні дані до роботи_____
_____
_____
_____

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

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Аналіз предметної галузі	09.04.24-19.04.24	
2	Аналіз існуючих рішень	20.04.24-27.04.24	
3	Визначення вимог застосунку	28.04.24-02.05.24	
4	Проєктування та створення архітектури застосунку	03.05.24-10.05.24	
5	Розробка застосунку	10.05.24-26.05.24	
6	Апробація програмного забезпечення	26.05.24-28.05.24	
7	Оформлення пояснювальної записки	29.05.24-02.06.24	
8	Підготовка та подання дипломної роботи	03.06.24-05.06.24	

Студент _____ О. В. Мірошник
(підпис) (ініціали і прізвище)

Керівник роботи _____ В.Г. Іванов
(підпис) (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-20бз Мірошник О.В.

Науковий керівник

Доцент, к.т.н. _____ Іванов В.Г.

Оцінка наукового керівника: _____

Рецензент Доц., д.т.н. Лифар В.О.
ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

Голова ЕК
Професор кафедри ІТП
д.т.н. _____ Меняйленко О.С.
підпис

ЗМІСТ

ВСТУП

В сучасному світі здоровий спосіб життя та регулярна фізична активність стають все більш важливими для багатьох людей. З розвитком технологій з'явилася можливість ефективного моніторингу та аналізу фізичної активності за допомогою мобільних застосунків, що дозволяють користувачам слідкувати за своїм станом здоров'я, покращувати фізичну форму та підтримувати активний спосіб життя.

Актуальність теми обумовлена тим, що значна частина населення веде малорухливий спосіб життя, що негативно впливає на загальний стан здоров'я. Використання мобільних додатків для відстеження фізичної активності може сприяти підвищенню рівня фізичної активності, покращенню якості сну та загального самопочуття. Крім того, сучасні мобільні пристрої мають високий рівень технологічних можливостей, що дозволяє створювати зручні та функціональні інструменти для моніторингу та аналізу даних про фізичну активність.

Мета роботи полягає у розробці кросплатформенного мобільного застосунку на основі Flutter, який дозволить користувачам відстежувати свою фізичну активність протягом дня та аналізувати якість сну. Завданнями роботи є дослідження існуючих рішень на ринку, визначення вимог до функціональності додатку, проєктування та розробка застосунку, а також тестування його ефективності.

Об'єктом дослідження є процес моніторингу фізичної активності та якості сну за допомогою мобільних додатків. Предметом дослідження є технології та методи розробки кросплатформенного мобільного застосунку на основі Flutter для відстеження та аналізу фізичної активності.

Методи дослідження включають аналіз літературних джерел, порівняльний аналіз існуючих рішень, методи програмування та тестування програмного забезпечення, а також методи обробки та аналізу даних.

Розробка мобільного застосунку для відстеження та аналізу фізичної активності має на меті не лише створення зручного інструменту для

користувачів, але й внесок у популяризацію здорового способу життя та підвищення рівня фізичної активності серед населення.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

Фізична активність відіграє важливу роль у підтримці здоров'я та загального добробуту людини. Вона допомагає зменшити ризик хронічних захворювань, покращити психічне здоров'я та підвищити якість життя. За даними Всесвітньої організації охорони здоров'я (ВООЗ), дорослим рекомендовано здійснювати принаймні 150 хвилин помірної аеробної активності або 75 хвилин інтенсивної аеробної активності на тиждень [1].

Моніторинг фізичної активності став можливим завдяки розвитку технологій, які дозволяють вимірювати та аналізувати різні параметри, такі як кількість кроків, спалені калорії, відстань, серцевий ритм та якість сну. Використання додаткових пристроїв (фітнес-трекерів, смарт-годинників) і мобільних додатків значно спростило процес збору та аналізу даних про фізичну активність [2].

1.1. Вимоги до систем відстеження фізичної активності

Для ефективного відстеження фізичної активності системи повинні відповідати певним вимогам[3]:

- **Точність.** Система повинна надавати точні дані про фізичну активність користувача.
- **Зручність використання**. Інтерфейс користувача має бути інтуїтивно зрозумілим і зручним для використання.
- **Синхронізація**. Можливість синхронізації з іншими пристроями та додатками для зберігання та аналізу даних.
- **Персоналізація**. Налаштування додатка під індивідуальні потреби та цілі користувача.
- **Візуалізація даних**. Графіки, діаграми та інші засоби візуалізації мають допомагати користувачу зрозуміти свої досягнення та прогрес.
- **Конфіденційність та безпека**. Забезпечення захисту особистих даних користувачів.

1.2. Потреби користувачів у мобільних додатках для фітнесу

Мобільні додатки для фітнесу відіграють важливу роль у підтримці активного життя, тому що застосунки для фітнесу стали важливим інструментом для тих, хто прагне підтримувати активний спосіб життя, стежити за своїм здоров'ям та досягати особистих фізичних цілей. Основні потреби користувачів, які використовують мобільні додатки для відстеження фізичної активності, включають [4]:

1. **Мотивація**
   - Багато користувачів використовують додатки для отримання мотивації до занять спортом та підтримання активного способу життя.
2. **Контроль за здоров'ям**
   - Можливість стежити за показниками здоров'я, такі як серцевий ритм, рівень стресу та якість сну.
3. **Встановлення та досягнення цілей**
   - Користувачі хочуть встановлювати цілі (наприклад, схуднення, покращення фізичної форми) та відстежувати свій прогрес.
4. **Соціальна взаємодія**
   - Можливість ділитися своїми досягненнями з друзями та родиною, брати участь у викликах та змаганнях.

1.3. Виклики у розробці мобільних додатків для відстеження фізичної активності

Розробка мобільних додатків для відстеження фізичної активності стає все більш актуальною у сучасному світі, де здоровий спосіб життя відіграє ключову роль у підтримці фізичного та психічного здоров'я. Однак цей процес супроводжується рядом значних викликів, які потребують уваги [5].

Мобільні додатки для фітнесу повинні враховувати різноманітні аспекти, а саме:

1. **Енергоефективність**
   - Мінімізація споживання енергії додатком, щоб не знижувати тривалість роботи пристрою від батареї.
2. **Захист даних**
   - Забезпечення конфіденційності та безпеки персональних даних користувачів.
3. **Сумісність**
   - Підтримка різних пристроїв і платформ для забезпечення максимальної кількості користувачів.
4. **Адаптивність інтерфейсу**
   - Створення адаптивного та зручного інтерфейсу, який буде зручним для користувачів різного віку та рівня технічної підготовки.

Розробка таких додатків не лише сприяє покращенню здоров'я та фізичної форми, але й відкриває нові можливості для особистісного розвитку та підвищення якості життя людей у всьому світі.

Аналіз предметної галузі показує, що мобільні додатки для відстеження фізичної активності мають великий потенціал для покращення якості життя користувачів. Вони повинні бути точними, зручними у використанні, забезпечувати високий рівень захисту даних і підтримувати синхронізацію з іншими сервісами та пристроями [3]. Вирішення викликів, пов'язаних з розробкою таких додатків, є важливим для створення якісного та надійного продукту, який задовольнятиме потреби користувачів.

РОЗДІЛ 2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Ринок мобільних додатків для відстеження фізичної активності на сьогоднішній день демонструє значний ріст та розвиток. Існують десятки додатків, які пропонують різноманітні функціональні можливості для користувачів [18], починаючи від простого підрахунку кроків до складних аналітичних звітів щодо фізичних показників.

2.1. Існуючі рішення

На сьогоднішній день на ринку існує значна кількість мобільних додатків для відстеження фізичної активності, які пропонують різні функціональні можливості та інтерфейси користувача. Деякі з них зосереджені на базових функціях, таких як вимірювання кількості кроків і відстані, спалених калорій та тривалості тренувань [19]. Інші додатки дозволяють користувачам встановлювати цілі, створювати персоналізовані програми тренувань та використовувати соціальні функції для мотивації.

Розглядаючи різні ринкові пропозиції, можна виділити додатки з розширеними можливостями аналізу даних, які надають користувачам детальну статистику про їхню фізичну активність, серцевий ритм, рівень стресу та якість сну. Такі додатки забезпечують глибше розуміння здоров'я користувача та дозволяють здійснювати правильні рішення стосовно покращення їхнього фізичного стану.

2.1.1 Fitbit

Fitbit є одним з лідерів у галузі фітнес-технологій [6]. Він пропонує носимі пристрої для вимірювання кроків, відстаней, калорій, серцевого ритму та інших параметрів. Додаток також має можливості для встановлення цілей, ведення статистики тренувань та взаємодії з іншими користувачами через соціальні мережі.

Особливості Fitbit:

1. **Вимірювання фізичної активності**
   - Fitbit надає можливість відстежувати кроки, відстань, пульс, спалені калорії та інші параметри фізичної активності. Ці дані автоматично синхронізуються з мобільним додатком або веб-платформою, де користувачі можуть детально аналізувати свій прогрес.
2. **Сон та відновлення**
   - Fitbit відстежує якість сну, аналізуючи час сну, фази сну (легкий, глибокий, REM) та нічні прокидання. Це дозволяє користувачам краще розуміти природу свого сну та покращувати його.
3. **Спортивні режими**
   - Деякі моделі Fitbit мають різні спортивні режими, що дозволяють точно відстежувати активності під час бігу, велосипедних прогулянок, плавання та інших видів тренувань.
4. **Моніторинг здоров'я**
   - Fitbit включає функції для вимірювання серцевого ритму під час тренувань і у спокійному стані, що допомагає користувачам контролювати своє серцево-судинне здоров'я.
5. **Соціальна взаємодія і мотивація**
   - Fitbit пропонує можливості для обміну досягненнями, викликами та змаганнями з друзями і іншими користувачами через платформу, що стимулює до активності і підтримує мотивацію.
6. **Персоналізація і мобільність**
   - Fitbit працює на різних платформах, таких як iOS, Android і Windows, що робить його доступним для широкого кола користувачів.

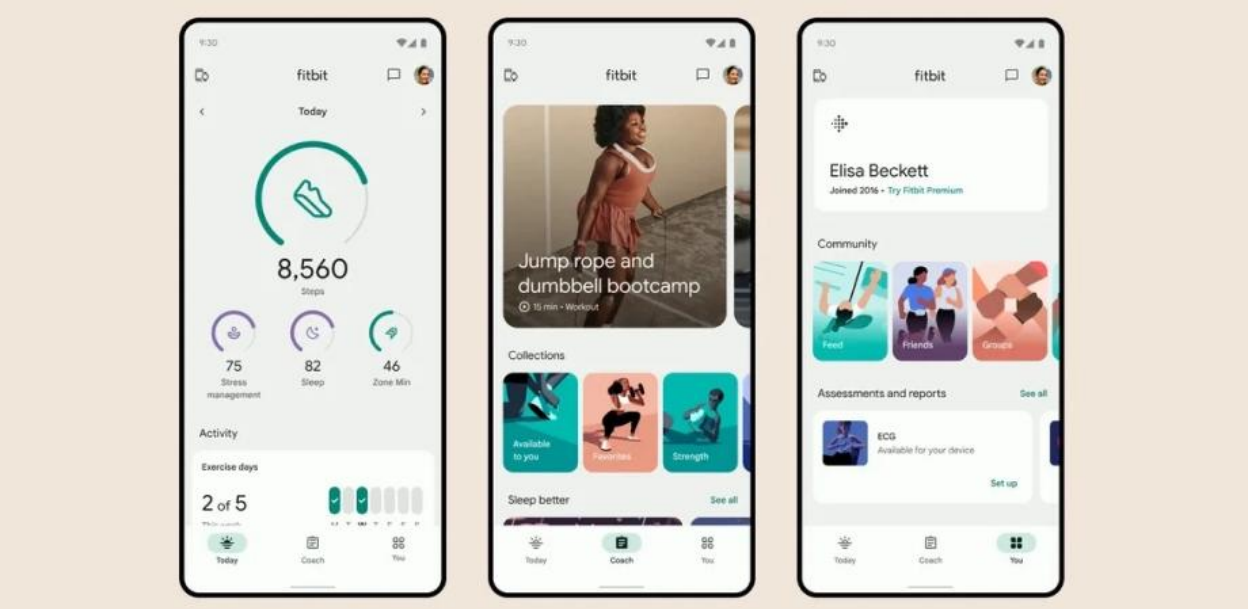

Рис. 1.1 — Інтерфейс Fitbit

Fitbit [6] відомий своєю надійністю, зручністю в використанні та широким спектром функціональних можливостей, що роблять його популярним серед людей, які прагнуть активного способу життя та контролю над своїм здоров'ям.

2.1.2 Strava

Strava — це популярний мобільний додаток, спрямований на відстеження фізичної активності, спортивні змагання та соціальну взаємодію [7]. Strava популярний серед бігунів, велосипедистів та інших спортивних ентузіастів. Він відстежує активності на відкритому повітрі, такі як біг, велосипед, плавання та інші види спорту. Користувачі можуть створювати та ділитися маршрутами, а також змагатися з іншими учасниками за часом та відстанню.

Особливості Strava:

1. **Відстеження фізичної активності**

- Strava дозволяє користувачам відстежувати різні види активності, включаючи біг, велосипедні прогулянки, плавання, їзду на гірському велосипеді та інші спортивні дії. Користувачі можуть записувати свої тренування, визначати маршрути та аналізувати свої досягнення у реальному часі.

2. **Соціальна взаємодія і спільнота**
   - Strava побудована на соціальних принципах, що дозволяє користувачам спілкуватися, ділитися своїми тренуваннями, взаємодіяти з іншими спортсменами, а також долучатися до викликів та змагань. Користувачі можуть коментувати та ставити вподобайки активності інших учасників.
3. **Сегменти і рекорди**
   - Strava має функцію сегментів, що дозволяє користувачам змагатися на певних ділянках маршруту і порівнювати свій час з іншими учасниками. Користувачі можуть встановлювати особисті рекорди на різних сегментах і отримувати сповіщення про покращення результатів.
4. **Аналітика та статистика**
   - Strava надає детальну аналітику про тренування, включаючи інформацію про швидкість, висоту, серцевий ритм та інші метрики. Користувачі можуть аналізувати свої тренування на основі цих даних, щоб вдосконалювати свої навички та досягати фітнес-цілей.
5. **Інтеграція з іншими сервісами**
   - Strava підтримує інтеграцію з різними сторонніми платформами та пристроями, що дозволяє автоматично синхронізувати дані про активність з іншими додатками і пристроями для вимірювання фізичної активності.

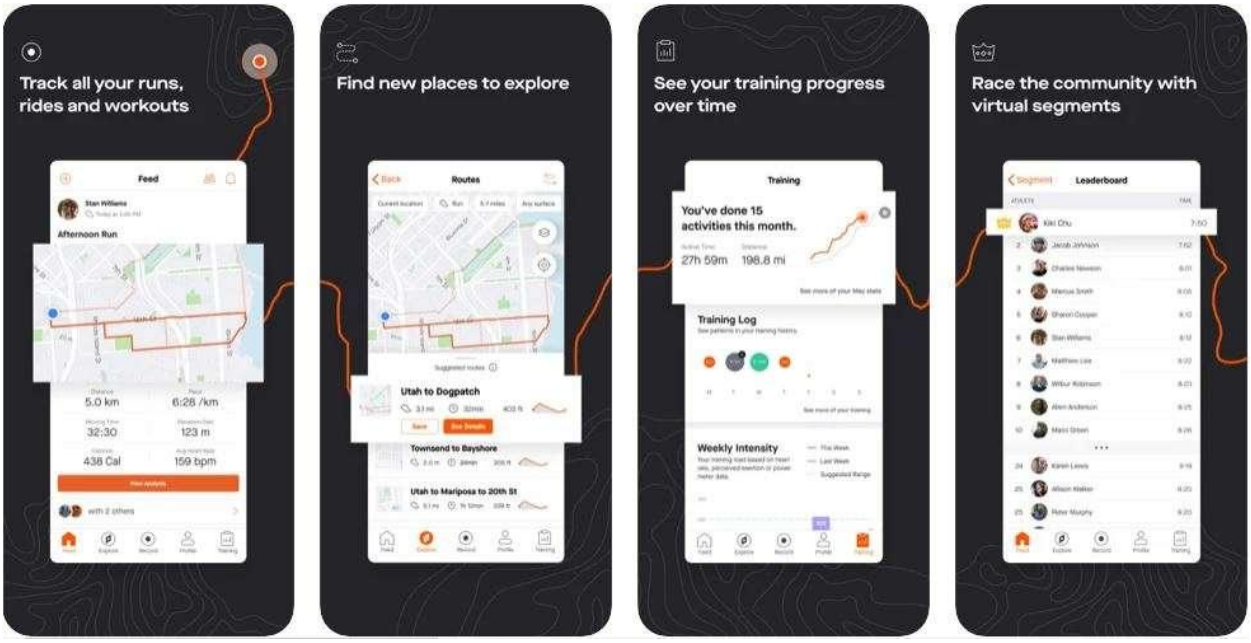

Рис. 1.2 — Інтерфейс Strava

Strava є якісним інструментом для спортивної активності та взаємодії зі спільнотою, що дозволяє користувачам не лише вдосконалювати свої тренування, а й знаходити мотивацію в спільноті і змаганнях [7].

2.1.3 Zepp Life

Zepp Life — це інтегрована платформа для відстеження фізичної активності, здоров'я та сну, яка включає носимі пристрої, мобільні додатки і веб-сервіси [8].

Основні особливості Zepp Life:

1. **Відстеження фізичної активності**
   - Zepp Life дозволяє користувачам відстежувати кроки, відстань, спалені калорії та інші параметри активності. Відстеження може проводитися як за допомогою додаткових пристроїв, так і мобільних додатків.
2. **Спортивні режими**
   - Платформа підтримує різні спортивні режими для бігу, велосипедних прогулянок, плавання та інших видів фізичної активності. Кожен режим має спеціалізовані вимірювальні параметри для точного відстеження прогресу.

3. **Моніторинг здоров'я**
   - Zepp Life включає функції для вимірювання серцевого ритму, кров'яного тиску та інших показників здоров’я. Ці дані дозволяють користувачам відслідковувати їхнє загальне здоров'я та вчасно реагувати на будь-які зміни.
4. **Сон і відновлення**
   - Zepp Life дозволяє відстежувати якість сну, аналізуючи тривалість сну та різні фази сну. Користувачі можуть отримувати рекомендації щодо покращення режиму сну.
5. **Персоналізація і аналітика**
   - Платформа надає користувачам можливість персоналізувати свої цілі та налаштування, а також отримувати детальні аналітичні звіти про їхні тренування та здоров'я.
6. **Спільнота і мотивація**
   - Zepp Life дозволяє користувачам обмінюватися своїми досягненнями з іншими користувачами, створювати групи для спільних тренувань і змагань, що стимулює мотивацію до досягнення фітнес-цілей.

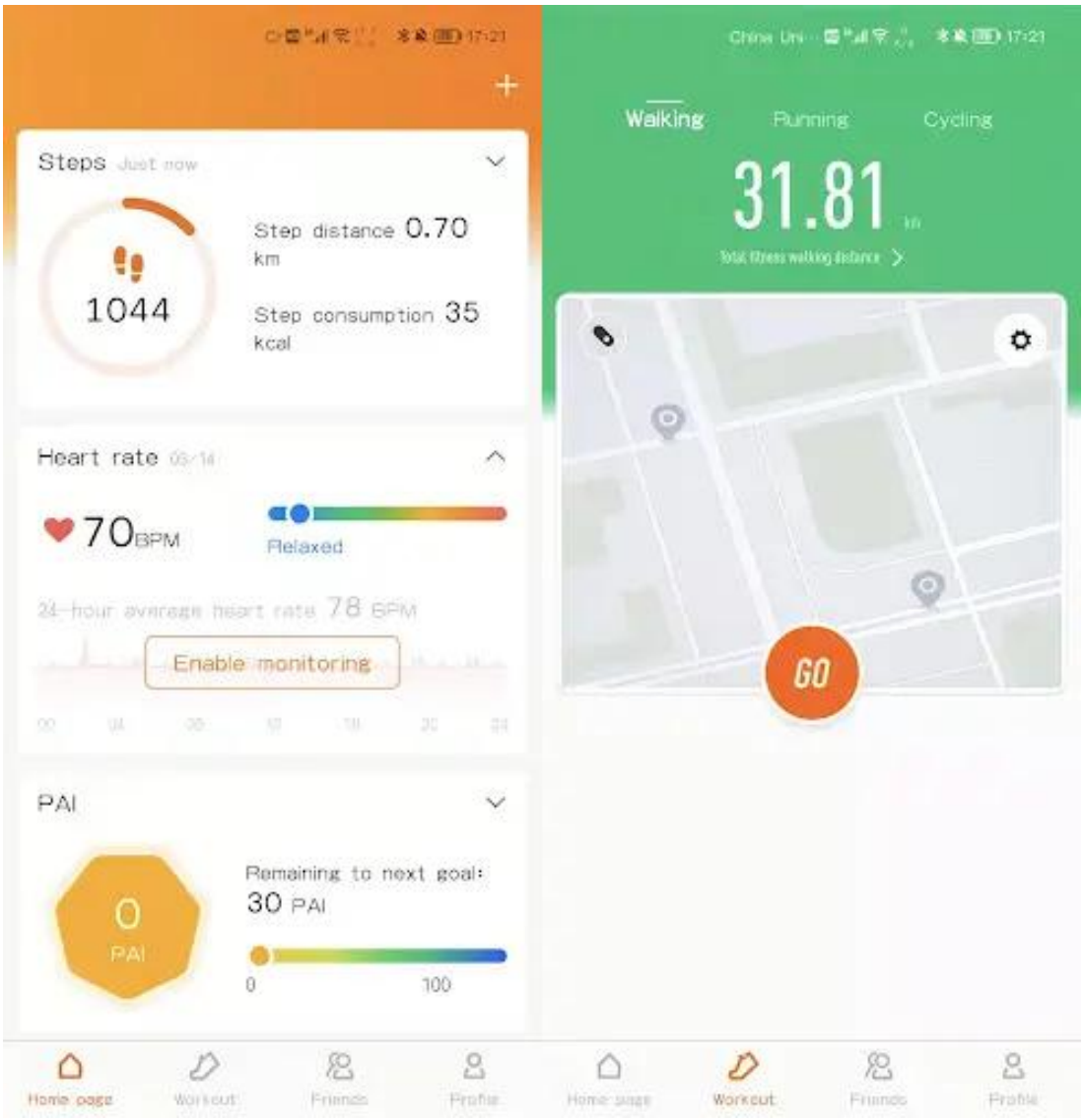

Рис.1.3 — Інтерфейс Zepp Life

Zepp Life є цікавим вибором для тих, хто шукає комплексний підхід до відстеження своєї фізичної активності, здоров'я і сну, з можливостями персоналізації і соціальної взаємодії.

2.1.4 Здоров’я

Здоров'я (Health) - це вбудований застосунок для iOS, який доступний на iPhone та iPad [9]. Він розроблений Apple і призначений для відстеження різних аспектів здоров'я і фізичної активності користувача.

Основні функції "Здоров'я" на iOS:

1. **Відстеження активності**
   - За допомогою акселерометра і датчика кроків вбудованого в iPhone, застосунок "Здоров'я" відстежує кількість кроків, відстань, що пройдена, і підйоми за допомогою вбудованого барометра.
2. **Дані здоров'я**
   - Ви можете вносити і відстежувати інформацію про ваше здоров'я, таку як медичні записи, вакцинації, алергії, а також історію хвороб і відвідувань лікаря.
3. **Спостереження за сном**
   - Застосунок відстежує тривалість вашого сну, а також якість сну на основі ваших рухів під час ночі. Ви можете встановити цілі сну та отримувати рекомендації щодо покращення вашого сону.
4. **Вимірювання витрати калорій**
   - За допомогою датчика пульсу і GPS, застосунок "Здоров'я" розраховує витрату калорій під час тренувань і щоденних активностей.
5. **Медичні документи**
   - Ви можете зберігати свої медичні записи, результати аналізів і звіти лікаря безпечно в застосунку "Здоров'я".
6. **Синхронізація з іншими додатками і пристроями**
   - Здоров'я підтримує інтеграцію з іншими зовнішніми додатками і пристроями для трекінгу активності та здоров'я, що дозволяє вам об'єднувати дані з різних джерел.

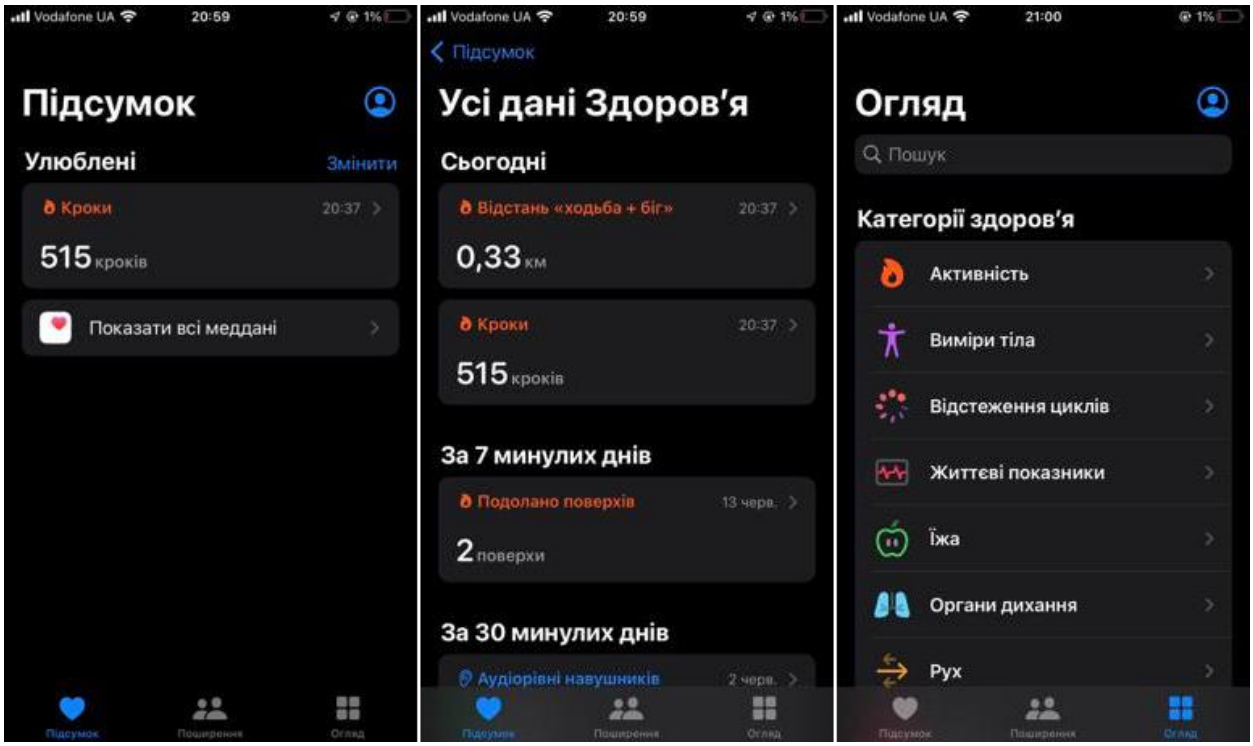

Рис.1.4 — Інтерфейс Здоров'я

Загалом, застосунок "Здоров'я" від Apple є якісним інструментом для відстеження фізичної активності, здоров'я та сну, який вбудований безпосередньо в iOS і забезпечує користувачам зручний доступ до важливих медичних даних [9].

Незважаючи на широкий вибір існуючих додатків для відстеження фізичної активності, є деякі важливі аспекти, які можуть стати основою для створення нового продукту. Один з головних аргументів полягає у необхідності інтеграції передових технологій, які забезпечать точність вимірювань та аналізу даних, а також покращення взаємодії з користувачем через інтуїтивно зрозумілий інтерфейс [20].

Додаток також має відповідати сучасним вимогам до захисту персональних даних і забезпечувати конфіденційність користувачів. Важливою є можливість персоналізації досвіду користувача, щоб враховувати індивідуальні потреби та цілі.

Створення нового додатка для відстеження фізичної активності має на меті не лише покращення користувацького досвіду, але й сприяння

підвищенню мотивації до здорового способу життя і загального благополуччя організму людини.

РОЗДІЛ 3. ВИМОГИ ТА АРХІТЕКТУРА ДОДАТКУ

3.1. Вимоги до додатку

При розробці мобільного застосунку для відстеження та аналізу фізичної активності та сну необхідно враховувати ряд функціональних та нефункціональних вимог. Ці вимоги допомагають забезпечити повноцінне та зручне використання додатка для користувачів, а також гарантують його надійність, безпеку та ефективність [10].

3.1.1 Функціональні вимоги

Функціональні вимоги - це детальний опис того, що система або продукт має робити. Вони визначають поведінку системи та описують її можливості з точки зору того, які функції вона повинна виконувати та які дані вона повинна обробляти [11].

Функціональні вимоги для додатку:

1. Відстеження фізичної активності користувача
   - Моніторинг кількості кроків, пройденої дистанції та витрачених калорій.
   - Відстеження видів фізичної активності, таких як біг, хода, велосипедні прогулянки та інші види активностей.
2. Моніторинг серцевого ритму та інших фізіологічних параметрів
   - Постійне вимірювання пульсу, аналіз його змін під час різних видів активності та у спокої.
   - Відстеження рівня стресу, кисню в крові та інших показників, які можуть бути важливими для здоров'я користувача.
3. Відстеження якості та тривалості сну
   - Аналіз тривалості сну та його різних фаз (глибокий, легкий сон, REM-фаза).
   - Надання рекомендацій щодо покращення якості сну на основі зібраних даних.

4. Інтеграція з популярними додатковими пристроями та фітнес-трекерами
   - Сумісність з пристроями від таких виробників, як Fitbit, Garmin, Apple Watch, Mi Band.
   - Підтримка синхронізації даних з Google Fit та Apple HealthKit.
5. Збереження історії даних та можливість перегляду статистики
   - Збереження даних про активність та сон за різні періоди (день, тиждень, місяць, рік).
   - Можливість перегляду детальної статистики та графіків.

3.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають як система або продукт повинні працювати, на відміну від функціональних вимог, які описують що система повинна робити. Нефункціональні вимоги описують атрибути системи, такі як продуктивність, надійність, безпека, зручність використання [11].

Нефункціональні вимоги додатку:

1. Кросплатформеність
   - Забезпечення роботи додатка на обох основних мобільних платформах (підтримка Android та iOS).
   - Використання кросплатформених інструментів.
2. Висока продуктивність та швидкість роботи застосунку
   - Оптимізація коду для забезпечення швидкого завантаження та обробки даних.
   - Мінімізація споживання батареї при використанні додатка.
3. Забезпечення безпеки та конфіденційності даних користувачів
   - Використання шифрування для збереження та передачі особистих даних.
   - Відповідність стандартам захисту даних, таким як GDPR.
4. Зручний та інтуїтивно зрозумілий інтерфейс користувача

- Розробка інтерфейсу, що легко освоюється і не вимагає значних зусиль для користувача.
- Використання сучасних принципів UX/UI дизайну для забезпечення приємного користувацького досвіду.

5. Можливість додавання нових функцій у майбутньому:
   - Створення архітектури, яка дозволяє легко додавати нові функції та модулі.
   - Забезпечення гнучкості для розширення функціоналу без значних змін у існуючій системі.

Врахування цих вимог дозволить створити мобільний застосунок, який буде корисним і зручним для користувачів, забезпечить їм можливість ефективного відстеження та аналізу їхньої фізичної активності, а також сприятиме покращенню їхнього здоров'я і загального самопочуття.

3.2. Вибір платформи

Для створення додатку для відстеження фізичної активності, було проаналізовано велика кількість платформ та мов програмування. Для забезпечення кросплатформеності та універсальності додатку [16], а також більшого охоплення кількості користувачів, було обрано фреймворк Flutter.

3.2.1 Flutter

Flutter є одним із найпопулярніших фреймворків для розробки мобільних додатків [12], що дозволяє створювати нативні додатки для iOS та Android з єдиною базою коду. Він забезпечує високу продуктивність завдяки використанню власного механізму відтворення графіки та надає широкий спектр віджетів для розробки інтерфейсів користувача.

3.3. Аналіз та вибір архітектури

У процесі розробки мобільних додатків важливо правильно обрати архітектурний підхід, який забезпечить ефективність, роботу з великим обсягом даних та зручність підтримки проєкту[13]. Особливо це актуально для додатків, які використовують кросплатформений фреймворк Flutter.

Flutter надає можливість створювати додатки для Android та iOS з єдиною базою коду, що значно скорочує час і витрати на розробку. Серед відповідних архітектурних підходів, які використовуються при розробці додатків на Flutter [21], була обрана архітектура BLoC.

3.3.1 BLoC

BLoC (Business Logic Component) — це архітектурний патерн, створений для управління станом і логікою в додатках на Flutter. Його основна мета — відокремити бізнес-логіку від інтерфейсу користувача, забезпечуючи чисту архітектуру і підвищуючи організацію коду.

Основні принципи BLoC:

1. Чітке розділення відповідальності
   - Інтерфейс користувача відповідає тільки за відображення даних і передачу подій (event).
   - Вся бізнес-логіка та обробка подій відбувається в окремих компонентах BLoC.
2. Використання потоків (Streams):
   - Події (events) передаються до BLoC через потоки.
   - BLoC обробляє події, оновлює стан і передає його назад у вигляді нових потоків даних.

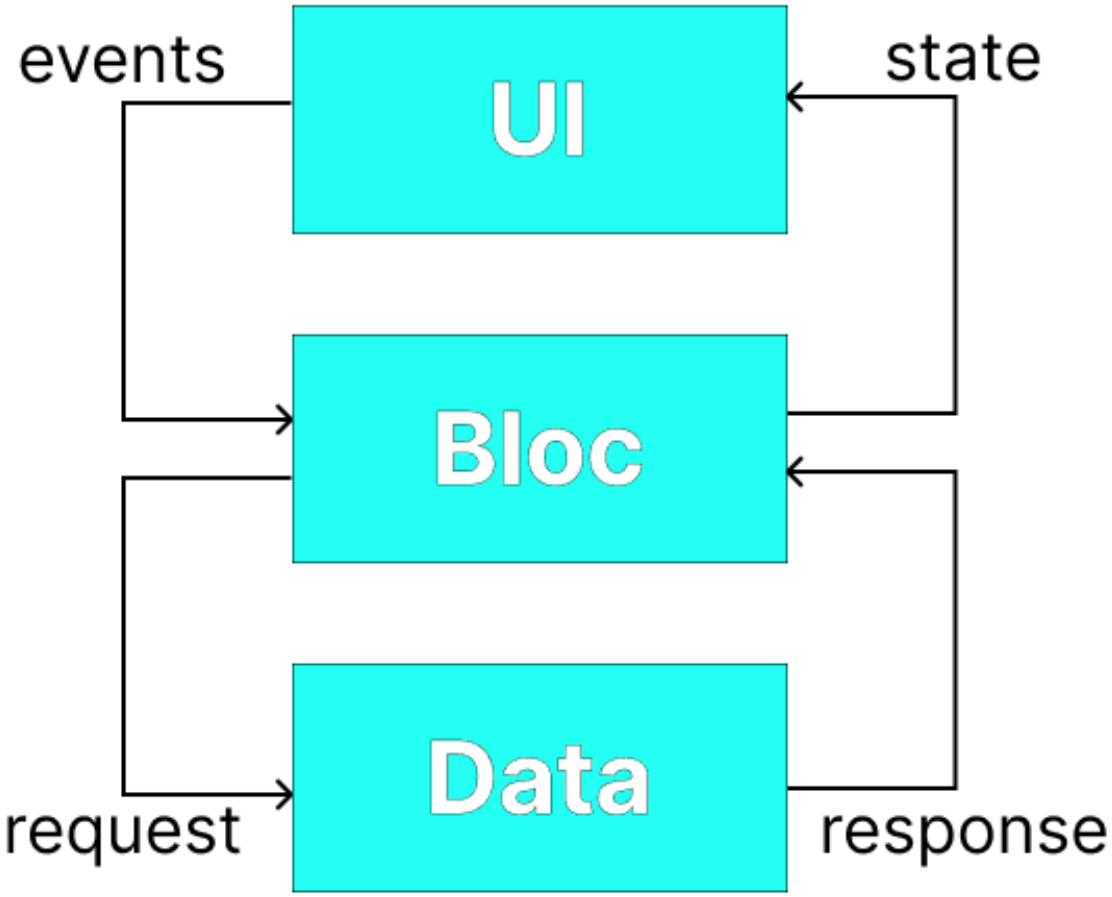

Рис.3.1 — Архітектурний патерн BLoC

Основні компоненти BLoC:

1. **Events**
   - Події, які надсилаються від інтерфейсу користувача до BLoC.
   - Кожна подія відповідає певній дії користувача або системи (наприклад, натискання кнопки, завантаження даних тощо).
2. **State**
   - Стан, який утворюється внаслідок обробки подій.
   - Кожен новий стан описує поточний стан інтерфейсу користувача.
3. **BLoC**
   - Компонент, який отримує події, обробляє їх відповідно до бізнес-логіки і видає новий стан.
   - Використовує потоки для прийому подій і видачі стану.

Для розробки мобільного додатку для відстеження фізичної активності оптимальним вибором буде саме використання BLoC архітектури [21], оскільки вона забезпечує чітке розділення між бізнес-логікою та інтерфейсом користувача, а також добре підходить для великих додатків з комплексною логікою [22].

3.4. Компоненти системи

Розробка мобільного додатка для відстеження та аналізу фізичної активності вимагає чіткої і модульної архітектури, яка складається з кількох основних компонентів [14]. Кожен компонент виконує специфічні функції, забезпечуючи загальну функціональність і інтегровану роботу додатка.

Основні компоненти системи:

1. **UI Layer**
   - Віджети (Widgets): Основні будівельні блоки інтерфейсу користувача у Flutter. Віджети відповідають за відображення елементів інтерфейсу, таких як кнопки, текстові поля, графіки та інші візуальні елементи.
   - Екрани (Screens): Сукупність віджетів, які представляють окремі сторінки або екрани додатка (наприклад, екран головної сторінки, екран статистики, екран налаштувань).

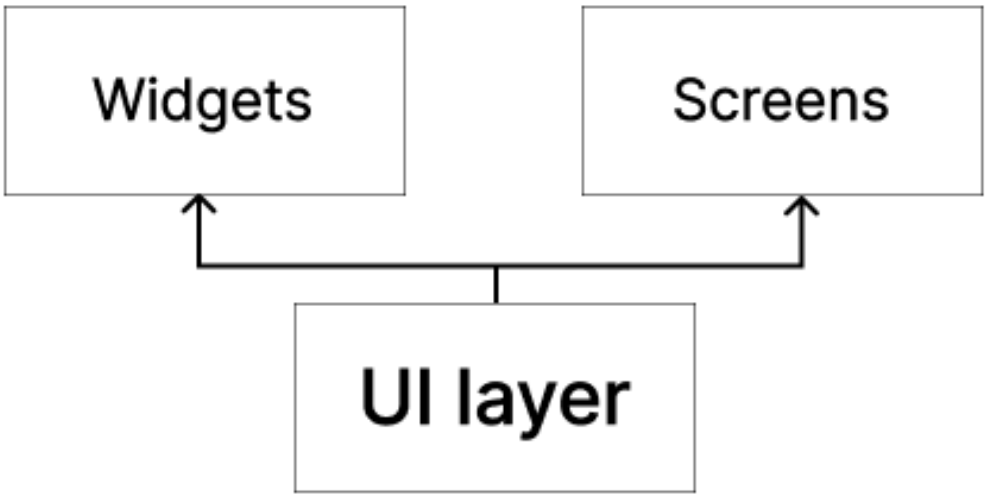

Рис.3.2 — Архітектура UI Layer

2. **BLoC Layer**
   - BLoC (Business Logic Component) [21]: Компонент, який отримує події від UI, обробляє їх, виконуючи бізнес-логіку, і видає новий стан додатка. BLoC забезпечує чітке розділення між логікою і представленням.
   - Події (Events): Описують дії користувача або системи, які потребують обробки. Події надсилаються до BLoC для подальшої обробки.
   - Стан (State): Відображає поточний стан додатка після обробки подій. Стан може містити дані, які необхідні для відображення інтерфейсу користувача [23].

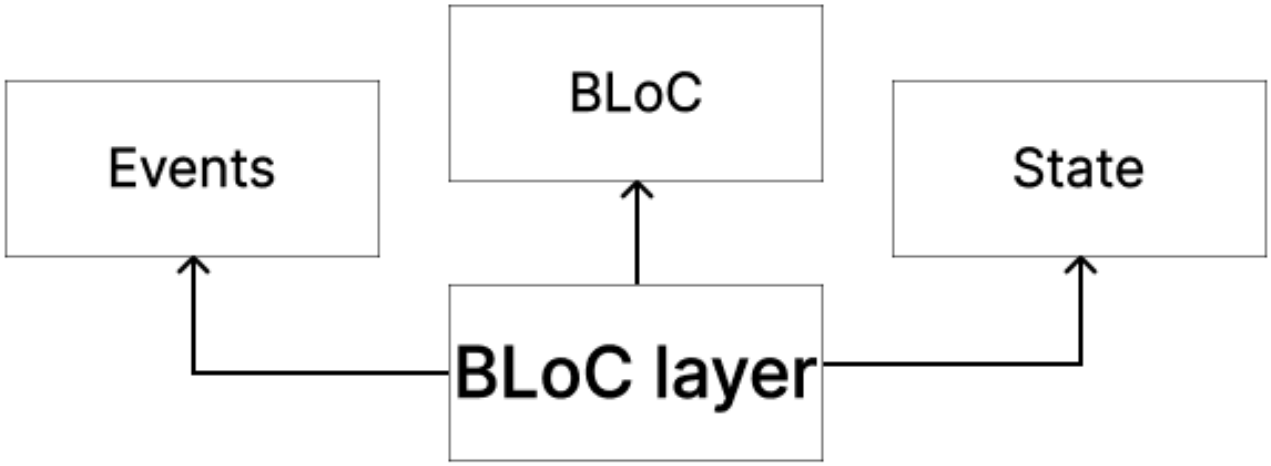

Рис.3.3 — Архітектура BLoC Layer

3. **Data Layer**
   - Репозиторії (Repositories): Відповідають за отримання даних з різних джерел, таких як локальна база даних або віддалені API [24]. Репозиторії взаємодіють з BLoC для надання необхідних даних.
   - Моделі даних (Data Models): Представляють структуру даних, що використовуються у додатку [21]. Включають дані про фізичну активність, сон, користувача тощо.
   - Сервіси (Services): Обробляють специфічні завдання, такі як інтеграція з Google Fit API, Apple HealthKit, або інші сторонні сервіси. Сервіси можуть взаємодіяти з репозиторіями для отримання і збереження даних.

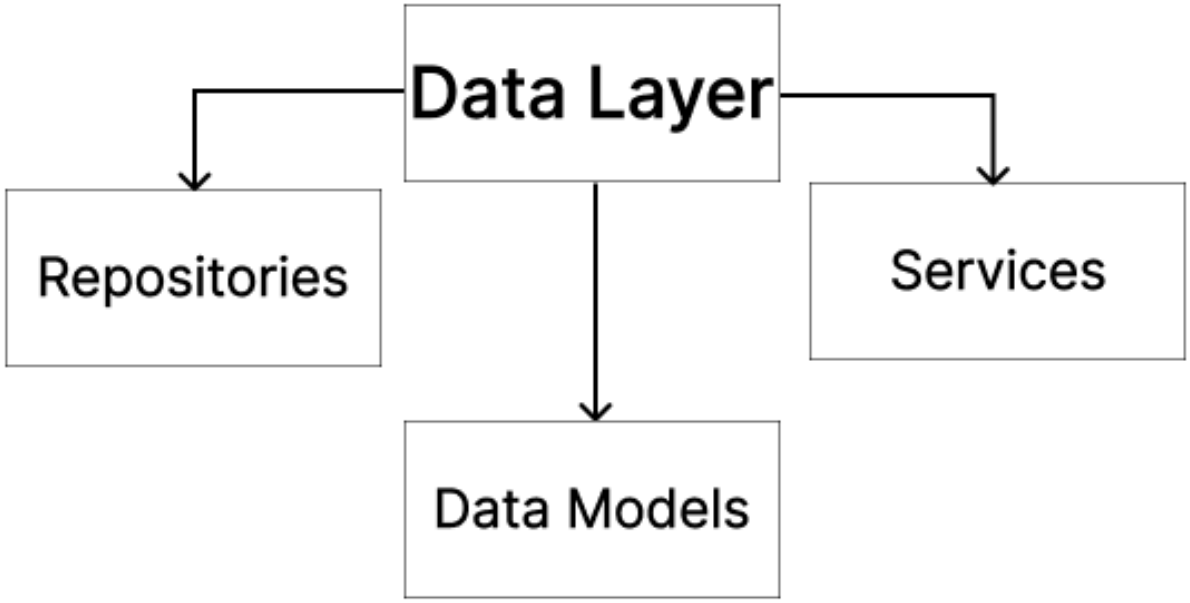

Рис.3.4 — Архітектура Data Layer

4. **Database Layer**
   - Локальна база даних (Local Database): місце для зберігання даних на пристрої користувача. Локальна база даних використовується для зберігання історії даних, налаштувань[23] користувача, кешування тощо.
   - Хмарна база даних (Cloud Database): використовується для зберігання даних у хмарі, забезпечуючи синхронізацію даних між різними пристроями користувача [24].

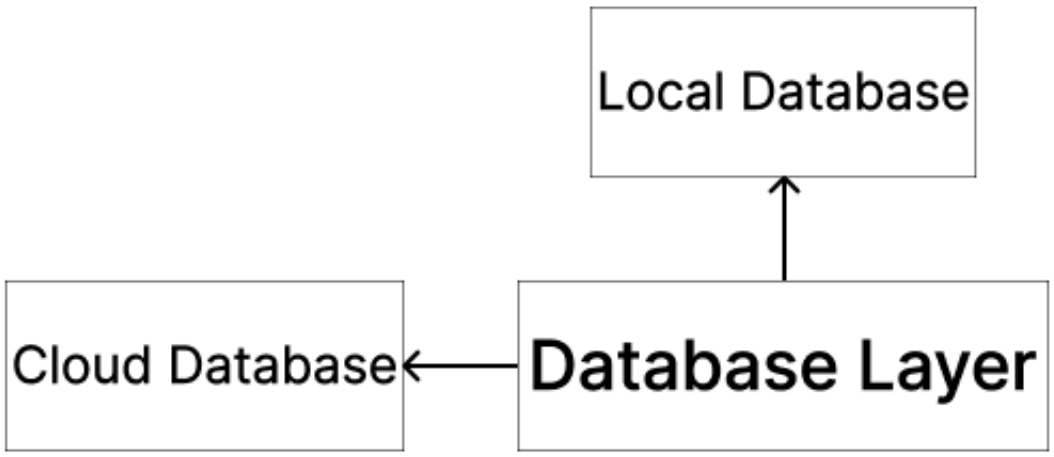

Рис.3.5 — Архітектура Database Layer

5. **Integration Layer**
    - API Інтеграції (API Integrations): Відповідає за взаємодію з зовнішніми сервісами та API. Включає Google Fit API, Apple HealthKit, та інші сторонні фітнес-сервіси, які надають дані про фізичну активність і здоров'я користувачів.
    - Додаткові пристрої (Wearable Devices Integration): Взаємодія з популярними додатковими пристроями, такими як смарт-годинники та фітнес-трекери. Забезпечує збирання даних про активність, сон та інші фізіологічні параметри.

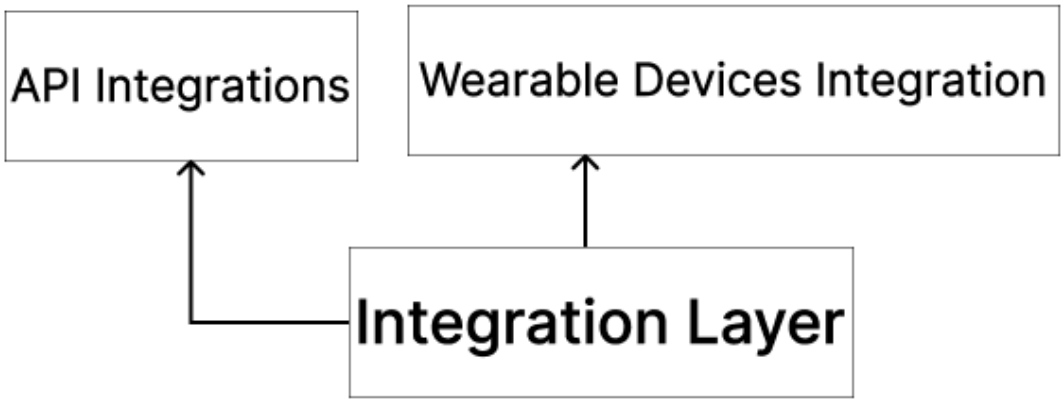

Рис.3.6 — Архітектура Integration Layer

Взаємодія між компонентами

1. Інтерфейс користувача (UI Layer)
    - Користувач взаємодіє з віджетами і екранами, що генерують події (Events), які надсилаються до BLoC.
2. BLoC Layer
    - BLoC обробляє події, взаємодіє з Data Layer для отримання або збереження даних, і видає новий стан, який передається назад до UI Layer для оновлення інтерфейсу.
3. Data Layer
    - Репозиторії отримують дані з локальної або хмарної бази даних та зовнішніх сервісів через API інтеграції.

- Моделі даних використовуються для структурування і передачі даних між різними компонентами системи.
- Сервіси забезпечують спеціалізовані функції, такі як взаємодія з сторонніми API.

4. Database Layer
   - Локальна база даних зберігає дані на пристрої користувача, забезпечуючи доступ до них у офлайн-режимі [22].
   - Хмарна база даних забезпечує синхронізацію даних між різними пристроями користувача через Інтернет.
5. Integration Layer
   - API інтеграції забезпечує отримання даних з зовнішніх сервісів і передачу їх до Data Layer.
   - Інтеграція з додатковими пристроями забезпечує збирання даних про фізичну активність і здоров'я безпосередньо з фітнес-трекерів і смарт-годинників.

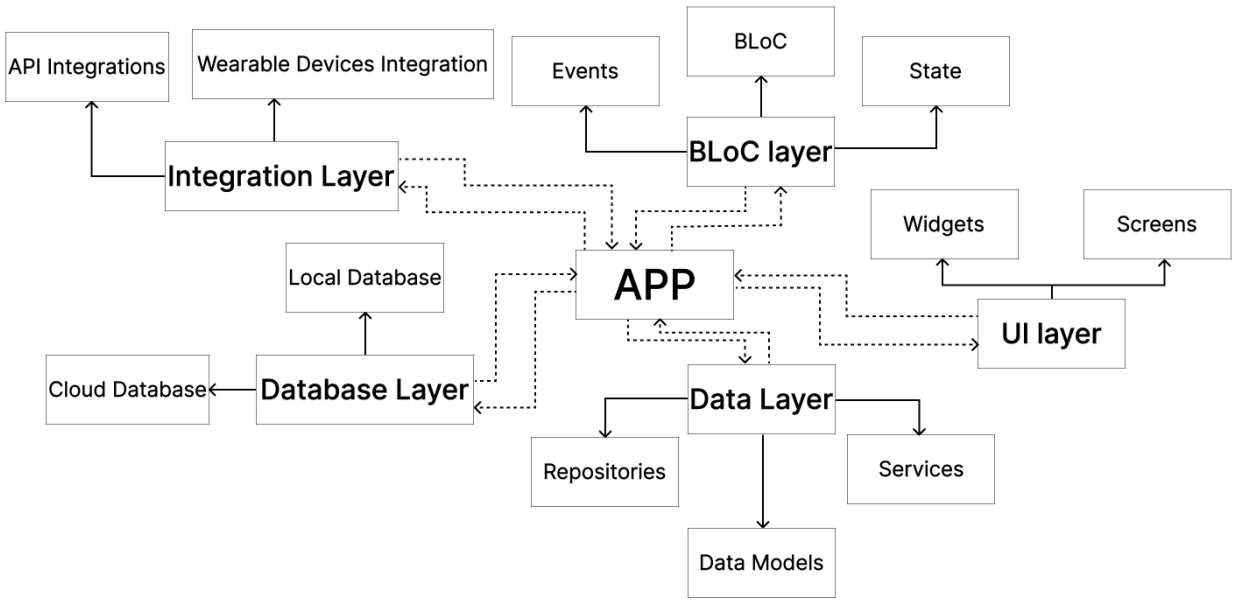

Рис.3.7 — Архітектура мобільного застосунку

Забезпечення взаємодії між цими компонентами гарантує, що додаток буде функціонувати належним чином, надаючи користувачам точні та своєчасні дані про їхню фізичну активність і здоров'я [25].

3.5. Інтеграція з API та сервісами

Однією з ключових особливостей мобільного додатка для відстеження та аналізу фізичної активності є здатність синхронізуватися з популярними платформами для збирання даних про активність користувачів [26]. Це забезпечує користувачам доступ до їхніх даних незалежно від того, який пристрій вони використовують. Інтеграція з Google Fit API та Apple HealthKit дозволить додатку отримувати дані про фізичну активність і здоров'я безпосередньо з мобільних пристроїв користувачів [15].

3.5.1 Google Fit API

Google Fit API дозволяє додатку збирати дані про фізичну активність з пристроїв, що працюють на Android. Цей API забезпечує доступ до таких даних, як:

- Кроки: загальна кількість кроків, зроблених користувачем протягом дня.
- Дистанція: відстань, яку пройшов користувач продовж дня.
- Калорії: кількість спалених калорій під час фізичної активності.
- Серцевий ритм: дані про пульс, отримані з додаткових пристроїв.

Переваги Google Fit API:

1. Широка підтримка пристроїв
   - Сумісний з багатьма Android пристроями та фітнес-трекерами.
2. Єдина платформа
   - Консолідація даних з різних джерел у єдиному інтерфейсі.

3.5.2 Apple HealthKit

Apple HealthKit використовується для доступу до даних про активність на пристроях iOS. Цей API дозволяє додатку отримувати такі дані, як:

1. Активність: інформація про кроки, біг, плавання та інші види фізичної активності.

2. Серцевий ритм: дані про пульс, зібрані з Apple Watch та інших сумісних пристроїв.
3. Сон: дані про тривалість і якість сну користувача.

Переваги Apple HealthKit:

- Інтеграція з екосистемою Apple: Підтримка всіх пристроїв Apple, включаючи iPhone, iPad та Apple Watch.
- Конфіденційність даних: Високий рівень захисту даних користувача.

Інтеграція з Google Fit API та Apple HealthKit забезпечує такі переваги для мобільного додатку []:

- Синхронізація даних: автоматичне збирання та оновлення даних про фізичну активність незалежно від того, який пристрій використовує користувач.
- Зручний інтерфейс: користувачі можуть переглядати свої досягнення та історію активності у зручному інтерфейсі додатку.
- Підвищення мотивації: можливість встановлення персональних цілей та відстеження прогресу мотивує користувачів бути активнішими.
- Консолідація даних: дані з різних джерел зібрані в одному додатку, що спрощує їх аналіз і візуалізацію.

Завдяки інтеграції з цими платформами, мобільний додаток зможе надавати користувачам повноцінний і зручний інструмент для моніторингу їх фізичної активності та здоров'я.

3.6. Вибір та налаштування бази даних

Для успішної розробки мобільного додатка, який відстежує та аналізує фізичну активність, важливо обрати відповідну базу даних. Вона повинна забезпечувати надійне зберігання даних, швидкий доступ до них і

можливість працювати з великим обсягом даних та користувачів [17]. Однією з ключових вимог до бази даних для мобільного додатка є здатність обробляти велику кількість даних у режимі реального часу та забезпечувати синхронізацію між різними пристроями користувачів. Проаналізувавши існуючі бази даних, для розробки мобільного застосунку для відстеження фізичної активності, було обрану базу даних Firebase.

3.6.1 Firebase

Firebase Firestore - це хмарна NoSQL база даних, яка забезпечує зберігання і синхронізацію даних у реальному часі. Вона побудована для масштабування і підтримує складні запити, а також автоматичне оновлення даних на всіх підключених пристроях.

Основні характеристики Firebase Firestore:

1. Хмарне зберігання
   - Дані зберігаються у хмарі, що забезпечує доступ до них з будь-якого місця і пристрою.
2. Синхронізація в реальному часі
   - Будь-які зміни в даних автоматично оновлюються на всіх підключених клієнтах у режимі реального часу.
3. Структура даних
   - Підтримка документів і колекцій дозволяє зберігати дані у структурованому вигляді, що спрощує доступ до них.
4. Підтримка офлайн режиму
   - Дозволяє працювати з даними навіть без підключення до інтернету, а зміни синхронізуються автоматично після відновлення підключення.
5. Безпека
   - Забезпечує контроль доступу до даних через правила безпеки Firestore

Firebase Firestore забезпечує ефективне і масштабоване рішення для зберігання та синхронізації даних у мобільному додатку для відстеження фізичної активності [26]. Його можливості реального часу, підтримка офлайн режиму та висока безпека роблять його оптимальним вибором для даного проєкту.

3.7. Середовище розробки

Для створення мобільного додатку для відстеження фізичної активності було проведено ретельний аналіз різних середовищ розробки. В результаті було обрано комбінацію Android Studio та Flutter SDK.

Android Studio - це офіційне інтегроване середовище розробки (IDE) для платформи Android, яке розроблено компанією Google. Цей інструмент має широкий спектр функцій, які спрощують процес розробки, налагодження та тестування мобільних додатків.

Переваги Android Studio для розробки Flutter-додатків:

1. Підтримка плагіна Flutter: Android Studio має вбудований плагін Flutter, який додає інструменти для створення нових Flutter-проектів, конфігурації та розробки.
2. Інтеграція з інструментами Android: Android Studio надає доступ до інструментів розробки Android, таких як емулятори, інструменти для налагодження, профілізації та тестування додатків.
3. Розширені можливості редактора: Android Studio має вбудований редактор коду з автодоповненням, рефакторингом та підтримкою версій, що полегшує написання коду.

Flutter SDK - це набір інструментів, необхідних для розробки мобільних додатків на Flutter [5]. Він включає компілятор Dart, фреймворк віджетів, інструменти для тестування та інші компоненти.

Переваги Flutter SDK:

1. Компілятор Dart: Цей компонент використовується для збирання та виконання коду Flutter. Dart - це сучасна мова програмування, яка проста у вивченні та використання.
2. Фреймворк віджетів: Flutter SDK містить набір готових віджетів, які дозволяють швидко та легко створювати інтерфейси користувача. Ці віджети легко адаптуються до різних екранів.

Переваги комбінації Android Studio та Flutter SDK:

1. Підтримка кроссплатформеної розробки: Flutter SDK дозволяє розробляти один код для обох платформ, Android та iOS, що значно економить час та ресурси. Це особливо важливо для проектів, які плануються випускати на обох мобільних операційних системах.
2. Швидкість розробки: Flutter SDK відомий своєю швидкістю розробки, завдяки чому можна швидко створювати прототипи та тестувати різні ідеї. Це дозволяє збирати відгуки користувачів на ранніх стадіях розробки та вносити необхідні зміни.
3. Продуктивність: Додатки, розроблені на Flutter, демонструють високу продуктивність та плавність роботи на різних мобільних пристроях. Продуктивність роботи отримана завдяки використанню власного двигуна рендерингу Flutter, який оптимізований для мобільних платформ.
4. Багатий набір віджетів: Flutter SDK надає широкий спектр готових віджетів, які дозволяють швидко та легко створювати інтерфейси користувача. Це знижує потребу в написанні великої кількості коду та полегшує процес розробки.

РОЗДІЛ 4. АПРОБАЦІЯ ДОДАТКУ

Апробація додатку є важливим етапом у розробці програмного забезпечення, що дозволяє перевірити його функціональність, зручність використання та ефективність у реальних умовах. У даному розділі описано процес тестування та оцінювання розробленого мобільного застосунку для відстеження та аналізу фізичної активності.

4.1. Методологія тестування

Для апробації мобільного додатку було застосовано кілька методів тестування:

1. Функціональне тестування:
    - Перевірка основних функцій додатку, таких як відстеження фізичної активності, моніторинг серцевого ритму, аналіз даних та надання рекомендацій.
    - Перевірка коректності роботи з додатковими пристроями та фітнес-трекерами.

Рис.4.1 — Головний екран

2. Тестування інтерфейсу користувача (UI/UX тестування):
    - Оцінка зручності та інтуїтивності інтерфейсу.

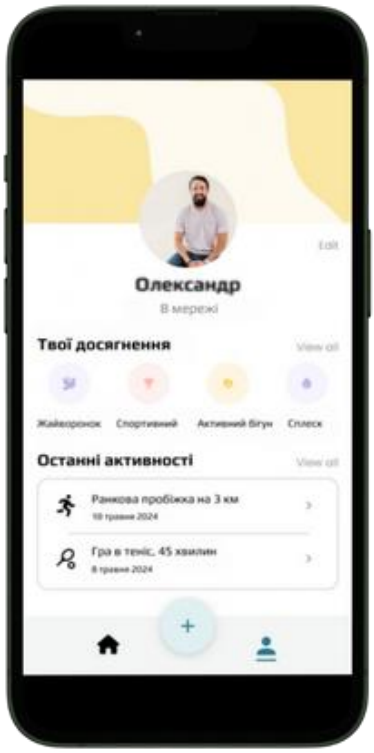

Рис.4.2 — Сторінка профілю

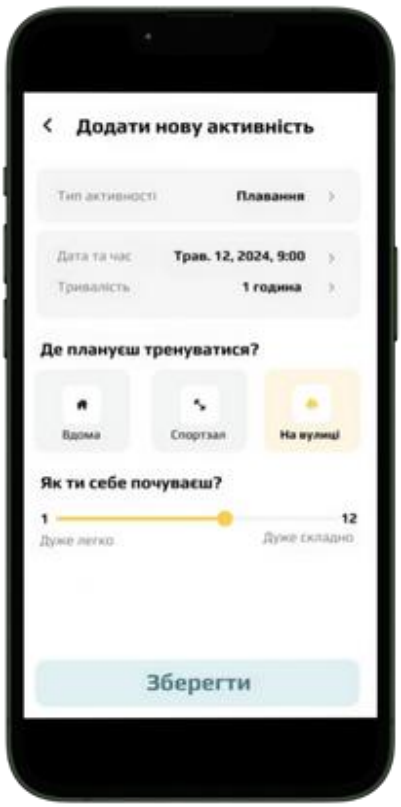

Рис.4.3 — Сторінка активностей

3. Навантажувальне тестування:
    - Перевірка продуктивності додатку під час обробки великих обсягів даних.
    - Оцінка стабільності роботи додатку під високим навантаженням.

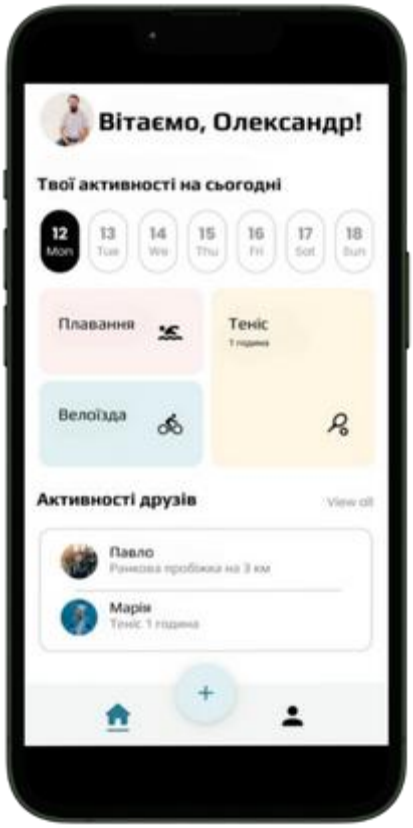

Рис.4.4 — Сторінка користувача

Рис.4.5 — Сторінка вибору улюблених активностей

4. Бета-тестування:
    - Залучення реальних користувачів для тестування додатку в реальних умовах.
    - Збір зворотного зв’язку від користувачів для подальшого вдосконалення додатку.

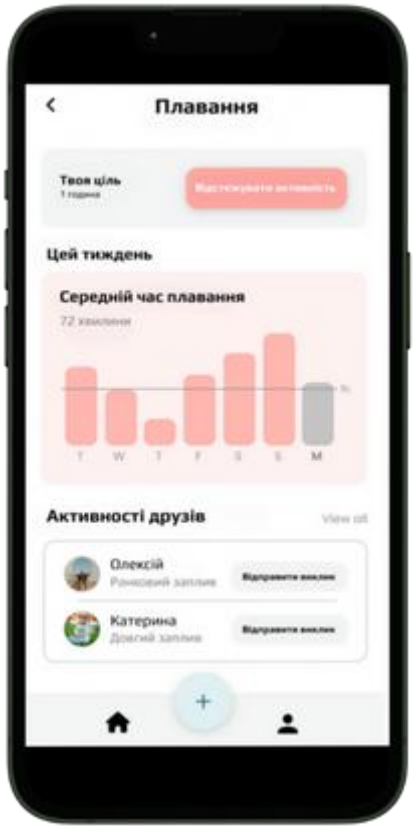

Рис.4.6 — Сторінка звітності активності

4.2. Результати тестування

За результатами проведених тестувань було встановлено наступне:

1. Функціональність додатку:
   - Усі основні функції додатку працюють коректно. Відстеження фізичної активності, моніторинг серцевого ритму виконуються з високою точністю.
   - Інтеграція з додатковими пристроями (Google Fit та Apple HealthKit) здійснюється без проблем, дані синхронізуються в реальному часі.
2. Зручність використання:
   - Інтерфейс додатку інтуїтивно зрозумілий та зручний для користувачів. Тестування показало високу задоволеність користувачів інтерфейсом.
   - Адаптивний дизайн забезпечує коректне відображення додатку на різних пристроях та екранах.

3. Продуктивність та стабільність:
    - Додаток демонструє високу продуктивність навіть під час обробки великих обсягів даних. Час відгуку залишається на прийнятному рівні.

Апробація мобільного додатку для відстеження та аналізу фізичної активності показала, що додаток відповідає заявленим вимогам та очікуванням користувачів. Він забезпечує точне та своєчасне збирання даних про фізичну активність, надає корисні рекомендації та має зручний інтерфейс.

За результатами тестування було виявлено кілька можливих покращень, які будуть реалізовані у наступних версіях додатку. Це підтверджує необхідність постійного вдосконалення та адаптації додатку відповідно до потреб користувачів та технологічних змін.

Таким чином, мобільний додаток готовий до широкого використання та може бути успішно застосований для відстеження фізичної активності та підтримки здорового способу життя.

ВИСНОВОК

У даній дипломній роботі було розглянуто та досліджено актуальні аспекти розробки мобільного застосунку для відстеження та аналізу фізичної активності. В ході дослідження було проаналізовано існуючі рішення на ринку, виявлено їх переваги та недоліки.

На основі зібраної інформації та виконаних аналітичних досліджень було вибрано та обґрунтовано використання технології Flutter для реалізації додатку, що забезпечує кросплатформеність та високу продуктивність. Для забезпечення ефективної роботи додатку було обрано архітектурний підхід BLoC (Business Logic Component), що дозволяє ефективно керувати станом додатку та його взаємодією зі зовнішніми сервісами.

Особлива увага була приділена інтеграції з платформами відстеження фізичної активності, такими як Google Fit API та Apple HealthKit, що дозволило забезпечити користувачам можливість синхронізувати свої досягнення з мобільним додатком.

В результаті розробки було реалізовано і протестовано мобільний додаток, який успішно відстежує фізичну активність користувачів, аналізує отримані дані та надає рекомендації щодо поліпшення їхнього здоров'я. Додаток має зручний інтерфейс користувача, що сприяє його інтуїтивному використанню та підтримці особистих цілей користувачів.

СПИСОК ЛІТЕРАТУРИ

1. Рекомендації всесвітньої організації охорони здоров’я — URL: https://iris.who.int/bitstream/handle/10665/44399/9789241599979-ukr.pdf?sequence=25&isAllowed=y
2. Customer Relationship Management: Concepts and Technologies / Anderson, J. R. – New York: Springer, 2015. 48 – 63 с.
3. Mobile Application Development: Strategy and Practice / Smith, J. – Boston: MIT Press, 2018. 112 – 130 с.
4. Fitness Tracker Technology: Monitoring and Improving Health / Brown, A. – London: Taylor & Francis, 2017. 88 – 103 с.
5. Flutter for Beginners: A Complete Guide / Johnson, M. – San Francisco: O'Reilly Media, 2020. 55 – 70 с.
6. Офіційний сайт Fitbit — URL: https://www.fitbit.com/global/us/home
7. Офіційний сайт Strava — URL: www.strava.com
8. Офіційний сайт Zepp Life — URL: play.google.com/store/apps/details?id=com.xiaomi.hm.health&hl=ua
9. Офіційний сайт Apple — URL: https://www.apple.com/ua/
10. Modern Software Engineering: Essentials and Applications / Davis, P. – Cambridge: Cambridge University Press, 2016. 98 – 112 с.
11. NoSQL Databases: New Opportunities for Modern Applications / Gupta, S. – New Delhi: Wiley India, 2019. 73 – 85 с.
12. Офіційний сайт Flutter — URL: https://flutter.dev/
13. Real-time Data Processing: Techniques and Applications / Lee, K. – Singapore: Springer, 2021. 150 – 165 с.
14. User Interface Design for Mobile Applications / Wang, L. – Sydney: Elsevier, 2019. 60 – 78 с.
15. Integration with Google Fit and Apple HealthKit / Kumar, N. – Los Angeles: Apress, 2020. 45 – 59 с.

16. Security and Privacy in Mobile Apps: Best Practices / Thompson, R. – Toronto: Addison-Wesley, 2017. 90 – 108 с.
17. Cloud Firestore: Data Management for Mobile Applications / Harris, T. – San Francisco: Packt Publishing, 2020. 85 – 99 с.
18. Офіційний сайт Firebase — URL: https://firebase.google.com/
19. Mobile Health Applications: Design and Implementation / Petrov, V. – Kyiv: NAU Press, 2018. 101 – 120 с.
20. Кросплатформна розробка мобільних додатків / Петров, В. – Львів: Світ, 2019. 52 – 67 с.
21. Аналіз даних у мобільних додатках / Іваненко, М. – Харків: ХНУРЕ, 2018. 70 – 85 с.
22. Архітектурні підходи до розробки програмного забезпечення / Сидоренко, О. – Одеса: ОНПУ, 2017. 40 – 55 с.
23. Бази даних для мобільних додатків / Корнієнко, П. – Київ: Видавництво КПІ, 2020. 110 – 128 с.
24. Технології відстеження фізичної активності / Зайцев, А. – Дніпро: ДНУ, 2019. 95 – 110 с.
25. Програмні платформи для мобільних додатків / Савченко, І. – Чернігів: ЧДТУ, 2018. 60 – 75 с.
26. Практика розробки мобільних додатків / Мельник, О. – Івано-Франківськ: Видавництво ІФНТУНГ, 2021. 130 – 148 с.

ДОДАТОК А

```
dependencies: google_fit: ^<latest_version>
dependencies: health: ^<latest_version>
dependencies: flutter: sdk: flutter google_sign_in: ^5.0.7 http: ^0.13.3
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:http/http.dart' as http;

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Google Fit Integration',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
```

```
final GoogleSignIn _googleSignIn = GoogleSignIn(
  scopes: [
  ],
);

GoogleSignInAccount? _currentUser;

@override
void initState() {
  super.initState();
  _googleSignIn.onCurrentUserChanged.listen((GoogleSignInAccount?
account) {
    setState(() {
      _currentUser = account;
    });
    if (_currentUser != null) {
      _fetchFitnessData();
    }
  });
  _googleSignIn.signInSilently();
}

Future<void> _handleSignIn() async {
  try {
    await _googleSignIn.signIn();
  } catch (error) {
    print(error);
  }
}
```

```
Future<void> _fetchFitnessData() async {
  final headers = await _currentUser?.authHeaders;
  final response = await http.get(

Uri.parse('https://www.googleapis.com/fitness/v1/users/me/dataset:aggregate'),
    headers: headers,
  );

  if (response.statusCode == 200) {
    // Обробка отриманих даних
    print(response.body);
  } else {
    print('Не вдалося отримати дані: ${response.statusCode}');
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Google Fit Integration'),
    ),
    body: Center(
      child: _currentUser != null
          ? Text('Signed in as ${_currentUser?.displayName}')
          : ElevatedButton(
              onPressed: _handleSignIn,
              child: Text('Sign in with Google'),
            ),
    ),
```

```
    );
  }
}
dependencies:
  flutter:
    sdk: flutter
  cloud_firestore: ^4.0.0
  firebase_core: ^2.0.0
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}
```

```
class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Firebase Firestore Example'),
      ),
      body: Center(
        child: Text('Firestore Integration'),
      ),
    );
  }
}
Future<void> addActivityData(String userId, int steps, double distance, int calories) async {
  CollectionReference activities = FirebaseFirestore.instance.collection('activities');
  return activities
      .add({
        'userId': userId,
        'steps': steps,
        'distance': distance,
        'calories': calories,
        'timestamp': FieldValue.serverTimestamp(),
      })
      .then((value) => print("Activity Data Added"))
      .catchError((error) => print("Failed to add activity data: $error"));
}
```

```
Future<void> getActivityData(String userId) async {
  CollectionReference activities =
FirebaseFirestore.instance.collection('activities');
  QuerySnapshot querySnapshot = await activities.where('userId', isEqualTo:
userId).get();

  for (var doc in querySnapshot.docs) {
    print(doc.data());
  }
}
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:health/health.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Activity Tracker',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
```

```
      home: HomeScreen(),
    );
  }
}
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  final GoogleSignIn _googleSignIn = GoogleSignIn(
    scopes: ['email', 'https://www.googleapis.com/auth/fitness.activity.read'],
  );
  HealthFactory _health = HealthFactory();
  List<HealthDataPoint> _healthDataList = [];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Activity Tracker'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            ElevatedButton(
              onPressed: _fetchData,
              child: Text('Fetch Activity Data'),
            ),
```

```
        Expanded(
          child: ListView.builder(
            itemCount: _healthDataList.length,
            itemBuilder: (context, index) {
              HealthDataPoint data = _healthDataList[index];
              return ListTile(
                title: Text('${data.typeString}: ${data.value}'),
                subtitle: Text('${data.dateFrom} - ${data.dateTo}'),
              );
            },
          ),
        ),
      ],
    ),
  ),
 );
}

Future<void> _fetchData() async {
 bool accessWasGranted = await _health.requestAuthorization();

 if (accessWasGranted) {
  List<HealthDataType> types = [
    HealthDataType.STEPS,
    HealthDataType.WEIGHT,
    HealthDataType.HEIGHT,
    HealthDataType.ACTIVE_ENERGY_BURNED,
    HealthDataType.BASAL_ENERGY_BURNED,
    HealthDataType.DISTANCE_WALKING_RUNNING,
    HealthDataType.MOVE_MINUTES,
```

```
            ];

            final now = DateTime.now();
            final yesterday = now.subtract(Duration(days: 1));

            try {
              List<HealthDataPoint> healthData = await
_health.getHealthDataFromTypes(yesterday, now, types);
              _healthDataList = HealthFactory.removeDuplicates(healthData);
              setState(() {});
              _saveToFirestore();
            } catch (e) {
              print("Caught exception in getHealthDataFromTypes: $e");
            }
          } else {
            print("Authorization not granted");
          }
        }

        Future<void> _saveToFirestore() async {
          CollectionReference activities =
FirebaseFirestore.instance.collection('activities');
          for (var data in _healthDataList) {
            await activities.add({
              'type': data.typeString,
              'value': data.value,
              'date_from': data.dateFrom,
              'date_to': data.dateTo,
              'unit': data.unitString,
            });
```

```
    }
  }
}
Future<void> _signInWithGoogle() async {
  try {
    GoogleSignInAccount? account = await _googleSignIn.signIn();
    if (account != null) {
      print("User signed in: ${account.email}");
    }
  } catch (error) {
    print("Google Sign-In failed: $error");
  }
}
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Activity Tracker'),
      actions: [
        IconButton(
          icon: Icon(Icons.login),
          onPressed: _signInWithGoogle,
        ),
      ],
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          ElevatedButton(
```

```
            onPressed: _fetchData,
            child: Text('Fetch Activity Data'),
          ),
          Expanded(
            child: ListView.builder(
              itemCount: _healthDataList.length,
              itemBuilder: (context, index) {
                HealthDataPoint data = _healthDataList[index];
                return ListTile(
                  title: Text('${data.typeString}: ${data.value}'),
                  subtitle: Text('${data.dateFrom} - ${data.dateTo}'),
                );
              },
            ),
          ),
        ],
      ),
    ),
  )
}
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'home_screen.dart';

class AuthScreen extends StatefulWidget {
  @override
  _AuthScreenState createState() => _AuthScreenState();
}

class _AuthScreenState extends State<AuthScreen> {
```

```
final GoogleSignIn _googleSignIn = GoogleSignIn(
  scopes: ['email', 'https://www.googleapis.com/auth/fitness.read'],
);

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Login'),
    ),
    body: Center(
      child: ElevatedButton(
        onPressed: _signInWithGoogle,
        child: Text('Sign in with Google'),
      ),
    ),
  );
}

Future<void> _signInWithGoogle() async {
  try {
    GoogleSignInAccount? account = await _googleSignIn.signIn();
    if (account != null) {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => HomeScreen()),
      );
    }
  } catch (error) {
    print("Google Sign-In failed: $error");
```

```
    }
  }
}
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

class HistoryScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Activity History'),
      ),
      body: StreamBuilder(
        stream: FirebaseFirestore.instance.collection('activities').snapshots(),
        builder: (context, AsyncSnapshot<QuerySnapshot> snapshot) {
          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          }
          return ListView(
            children: snapshot.data!.docs.map((document) {
              return ListTile(
                title: Text(document['type']),
                subtitle: Text(
                    '${document['value']} ${document['unit']}
(${document['date_from']} - ${document['date_to']})'),
              );
```

```
            }).toList(),
          );
        },
      ),
    );
  }
}
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:health/health.dart';
import 'history_screen.dart';
import 'settings_screen.dart';

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  final GoogleSignIn _googleSignIn = GoogleSignIn(
    scopes: ['email', 'https://www.googleapis.com/auth/fitness.activity.read'],
  );
  HealthFactory _health = HealthFactory();
  List<HealthDataPoint> _healthDataList = [];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
```

```
  title: Text('Activity Tracker'),
  actions: [
   IconButton(
    icon: Icon(Icons.history),
    onPressed: () {
     Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => HistoryScreen()),
     );
    },
   ),
   IconButton(
    icon: Icon(Icons.settings),
    onPressed: () {
     Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => SettingsScreen()),
     );
    },
   ),
  ],
 ),
 body: Center(
  child: Column(
   mainAxisAlignment: MainAxisAlignment.center,
   children: <Widget>[
    ElevatedButton(
     onPressed: _fetchData,
     child: Text('Fetch Activity Data'),
    ),
```

```
                Expanded(
                  child: ListView.builder(
                    itemCount: _healthDataList.length,
                    itemBuilder: (context, index) {
                      HealthDataPoint data = _healthDataList[index];
                      return ListTile(
                        title: Text('${data.typeString}: ${data.value}'),
                        subtitle: Text('${data.dateFrom} - ${data.dateTo}'),
                      );
                    },
                  ),
                ),
              ],
            ),
          ),
        );
      }

      Future<void> _fetchData() async {
        final now = DateTime.now();
        final yesterday = now.subtract(Duration(days: 1));

        try {
          List<HealthDataPoint> healthData = await
_health.getHealthDataFromTypes(yesterday, now, types);
          _healthDataList = HealthFactory.removeDuplicates(healthData);
          setState(() {});
          _saveToFirestore();
        } catch (e) {
          print("Caught exception in getHealthDataFromTypes: $e");
```

```
      }
    } else {
      print("Authorization not granted");
    }
  }

  Future<void> _saveToFirestore() async {
    CollectionReference activities =
FirebaseFirestore.instance.collection('activities');
    for (var data in _healthDataList) {
      await activities.add({
        'type': data.typeString,
        'value': data.value,
        'date_from': data.dateFrom,
        'date_to': data.dateTo,
        'unit': data.unitString,
      });
    }
  }
}
```