

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ МЕТОДІВ ТА РОЗРОБЛЕННЯ ЗАСОБІВ
ВИЗНАЧЕННЯ РУХУ ДЛЯ МОБІЛЬНИХ ЗАСТОСУВАНЬ

(назва теми)

RESEARCH OF METHODS AND DEVELOPMENT OF MOTION
DETECTION TOOLS FOR MOBILE APPLICATIONS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-213м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

ЖИР Є.С.

(ПРІЗВИЩЕ та ініціали)

Керівник ТАБУНЩИК Г.В.

(ПРІЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРІЗВИЩЕ та ініціали)

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н., проф.
Сергій СУББОТІН
« » 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЖИР Єгора Сергійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження методів та розроблення засобів визначення руху для мобільних застосувань Research of Methods and Development of Motion Detection Tools for Mobile Applications

керівник проєкту (роботи) к.т.н., професор, ТАБУНЩИК Галина Володимирівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «07» жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи)) 2 грудня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір програмних засобів.

3. Розробка програми. 4. Експлуатація, тестування та результати.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1–4 Основна частина	ТАБУНЩИК Г.В., професор		
Нормоконтроль	БСЛОВА А.В., асистент		

7. Дата видачі завдання « 4 » вересня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи та підбір літератури.	1 тиждень	Розділ 1
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Вибір програмних засобів	4-5 тижні	Розділ 2
4	Розробка додатка	6-8 тижні	Розділ 3
5	Тестування	9 тиждень	Розділ 4
6	Формування висновків та написання пояснювальної записки	10-11 тижні	Додатки
7	Нормоконтроль та рецензування.	12 тиждень	
8	Захист роботи	12 тиждень	

Студент(ка)

_____ Єгор ЖИР
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Галина ТАБУНЩИК
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
214 с., 15 табл., 47 рис., 2 дод., 19 джерел.

ВИЗНАЧЕННЯ РУХІВ, ANDROID ЗАСТОСУНОК, KNN, КРОКИ, АКСИЛПРОМЕТР, ФІЛЬТР НИЖНІХ ЧАСТОТ, ЛОКОМОЦІЇ, ПЕДОМЕТР, ФІТНЕС.

Об'єкт дослідження – засоби визначення руху.

Предмет дослідження – дослідження методів та розроблення засобів визначення руху для мобільних застосувань.

Мета роботи полягає у відстеженні фізичної активності користувача в застосунку.

Матеріали, методи та технічні засоби: розробка Android застосунків, мовб програмування Kotlin і Python, середа розробки Android Studio і Jupiter Notebook, інтерфейс користувача написаний на Jetpack Compose, пристрій Samsung A15.

Результати: розроблено Android застасунок для підрахунку кроків і визначення руху користувача з використанням Kotlin, що використовує алгоритм KNN.

Висновки: проаналізовано та розроблено Android застасунок, який може відстежувати активність користувача.

Галузь використання – користувачі смартфонів на базі операційної системи Android.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 214 pages, 15 tables, 47 figures, 2 appendices, 19 sources.

MOTION DETECTION, ANDROID APPLICATION, KNN, STEPS, ACCELEROMETER, LOW-PASS FILTER, LOCOMOTION, PEDOMETER, FITNESS.

Research object – methods of motion detection.

Research subject – research of methods and development of motion detection tools for mobile application.

The purpose of the thesis is to track the user's physical activity in the application.

Materials, methods and techniques means: Android application development, Kotlin and Python programming languages, Android Studio and Jupiter Notebook development environment, user interface written in Jetpack Compose, Samsung A15 device.

Results: an Android application for counting steps and detecting user movement was developed using Kotlin, which utilizes the KNN algorithm.

Conclusions: analyzed and developed an Android application that can track user activity.

Field of use – users of smartphones based on the Android operating system.

ЗМІСТ

	С.
Перелік скорочень.....	9
Вступ.....	10
1 Аналіз предметної області.....	11
1.1 Аналіз існуючих застосувань, що вирішують аналогічні проблеми	11
1.1.1 Застосунок «Google Fit»	11
1.1.2 Застосунок «Samsung Health».....	12
1.1.3 Висновок аналізу застосунків.....	12
1.2 Аналіз і визначення локомоцій.....	13
1.3 Збір даних.....	16
1.4 Перетворення даних.....	18
1.5 Ознаки кроків	20
2 Вибір предметної області.....	25
2.1 Методи фільтрації даних	25
2.2 Метод класифікації даних	25
2.3 Вибір мови для написання тестової моделі.....	27
2.4 Вибір середовища розробки для написання тестової моделі	28
2.5 Мови Kotlin та Java: особливості та порівняння.....	28
2.6 Мови Kotlin та Groovy для Gradle: особливості та порівняння.....	29
2.7 Вибір бібліотеки для зберігання даних.....	30
2.8 Вибір бібліотеки для впровадження залежностей (DI)	31
2.9 Рішення щодо отримання кроків	32
3 Розробка програми.....	34
3.1 Аналіз вимог до застосування	34
3.1.1 Концепція.....	34
3.1.2 Функціональні вимоги до системи.....	34
3.1.3 Специфікація прецеденту «Відстеження інформації о локомоції»..	35
3.1.4 Специфікація прецеденту «Додавання кроків».....	36

3.2 Допоміжний застосунок	37
3.2.1 Інтерфейс допоміжного застосунку.....	38
3.2.2 Модель даних допоміжного застосунку.....	39
3.2.3 Збирання даних аксілірометра.....	40
3.3 Архітектура і структура додатка	41
3.3 Імплементація патерна проєктування	43
3.4 Проєктування інтерфейсу.....	45
3.4.1 Дизайн система і локалізація.....	45
3.4.2 Розбір екранів.....	46
3.5. Проєктування даних.....	54
3.5.1 Огляд шару data.....	54
3.5.2 Огляд шару domain.....	55
3.6 Збір даних.....	57
3.7 Перетворення даних.....	59
3.7.1 Агрегація зібраних даних.....	59
3.7.2 Перетворення в загальну величину прискорення.....	60
3.7.3 Фільтрація шумів.....	61
3.7.4 Фільтрація вершин та западин.....	62
3.7.5 Фільтрація локальних вершин та западин.....	63
3.7.6 Знаходження істинної вершини та западини.....	64
3.7.7 Трансформація даних в ознаки.....	66
3.8 Розпізнавання локомоцій	67
3.9 Інтеграція класифікатора з Android.....	72
4 Експлуатація, тестування та результати.....	73
4.1 Створення захищеного APK	73
4.2 Експлуатація додатка.....	76
4.3 Тестування додатка.....	77
4.3.1 Тестування сервісу переднього плану.....	77
4.3.2 Тестування точності зчитування кроків.....	78
4.3.2 Тестування точності класифікації кроків.....	80

Висновки	82
Перелік джерел і посилання.....	83
ДОДАТОК А Код програми.....	85
ДОДАТОК Б Слайди презентації	206

ПЕРЕЛІК СКОРОЧЕНЬ

API	– Application Programming Interface;
APK	– Android Application Package;
DI	– Dependency Injection;
DSL	– Domain-Specific Language;
IDE	– Integrated Development Environment;
JVM	– Java Virtual Machine;
KNN	– K-Nearest Neighbors;
MVC	– Model-View-Controller;
MVI	– Model-View-Intent;
MVP	– Model-View-Presenter;
MVVM	– Model-View-ViewModel;
SDK	– Software Development Kit;
UI	– User Interface;
XML	– Extensible Markup Language;
ООП	– Об’єктно Орієнтоване Програмування.

ВСТУП

Ми звикли завжди носити телефон із собою, куди б не вирушили. Постійна присутність пристрою поруч із користувачем, разом із широкими функціональними можливостями його, відкриває великі перспективи для відстеження фізичної активності користувача, що є важливим аспектом, як для спортсменів, які прагнуть ефективно планувати та відстежувати свої тренування, так і для пацієнтів, що проходять реабілітацію після травм.

Об'єктом дослідження є методи визначення руху.

Предмет дослідження – дослідження методів та розроблення засобів визначення руху для мобільних застосувань.

Мета роботи полягає у відстеженні фізичної активності користувача в застосунку.

Для досягнення поставленої мети у роботі вирішувалися такі завдання:

- дослідити якими способами можна здобути кроки користувача;
- вивчити як класифікувати фізичні активності (ходьба, біг підтюпцем і звичайний біг) для кроків;
- розробити нативний Android застосунок який здатний відображати фізичну активність користувача;
- оформити пояснювальну записку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих застосувань, що вирішують аналогічні проблеми

1.1.1 Застосунок «Google Fit»

Google Fit [1] – це застосунок для відстеження здоров'я та фізичної активності. Він дозволяє користувачам моніторити свої показники фітнес даних, такі як кроки, калорії, фізична активність та інші (рис. 1.1).

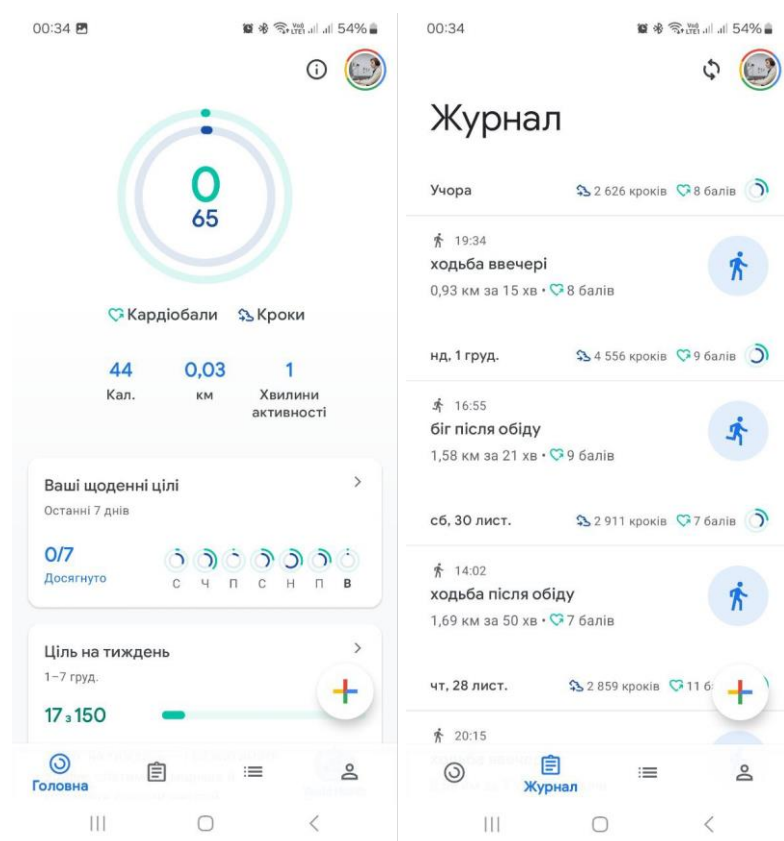


Рисунок 1.1 – Знімки екрана з функціональністю додатка Google Fit

Для розпізнавання фітнес активностей Google використовує сенсори телефону: акселірометр, гіроскоп та інші. Стратегія розпізнавання рухів – вибір найпоширенішого виду активності.

1.1.2 Застосунок «Samsung Health»

Samsung Health [2] – це схожий застосунок на Google Fit, який на відміну надає можливість переглядати фітнес відео, і не має можливості автоматичного розпізнавання фізичної активності – тільки обирати фізичну активність вручну (рис. 1.2)

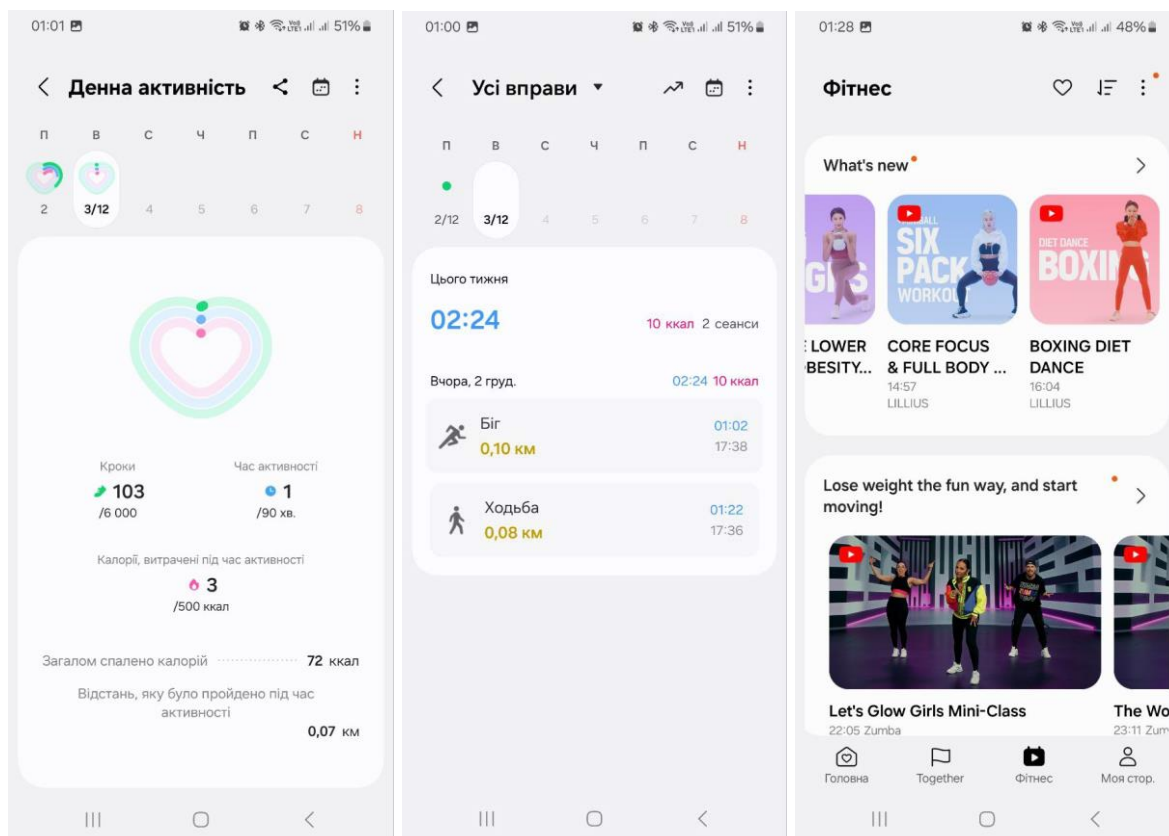


Рисунок 1.2 – Знімки екрана з функціональністю додатка Samsung Health

1.1.3 Висновок аналізу застосунків

Google Fit і Samsung Health надають спектр зручних і корисних можливостей для відстеження фітнес метрик і управління ними. Обидва застосунки збирають фітнес метрики за допомогою таких сенсорів телефону як аксилірометр і гіроскоп. Аксилірометр використовується для вимірювання

лінійного прискорення телефону за трьома осями X, Y і Z. Гіроскоп зі свого боку використовується для вимірювання кутового прискорення [3].

1.2 Аналіз і визначення локомоцій

У цьому розділі буде розглянуто характеристики локомоцій, зокрема ходьби, бігу підтюпцем і звичайного бігу. Хоча кожна людина має індивідуальні особливості рухів, обумовлені анатомічними та фізіологічними чинниками, можна виділити загальні патерни, характерні для більшості людей.

Рух людини має безліч властивостей, які можна згрупувати таким чином:

- циклічність – під час руху, нога виконує періодичний алгоритм дій;
- довжина кроку – різниця положення ноги на початку кроку і в кінці;
- висота кроку – висота підняти ноги;
- торкання ступи – торкання частиною ступні опорної поверхні;
- тривалість торкання – тривалість торкання ступні опорної поверхні;
- частота кроків – інтенсивність руху;
- положення корпусу – характеристика положення корпусу під час руху;
- амплітуда руки – описує розмах і амплітуду рухів руки під час кроків.

Ходьба – це природний спосіб пересування людини, за якого тіло переміщується за допомогою чергування рухів ніг, завжди зберігаючи контакт хоча б однієї ноги з поверхнею. Цей вид локомоції характеризується вертикальним положенням корпусу. Відбувається повний переكات стопи від п'яти до носка, амплітуда рухів рук невисока і синхронізована з кроками. Ходьба має помірну довжину і частоту кроків (рис. 1.3).



Рисунок 1.3 – Механіка ходьби

Звичайний біг – це спосіб швидкого пересування людини, за якого тіло переміщується завдяки чергуванню рухів ніг із періодичними фазами польоту, коли жодна нога не торкається землі. Для звичайного бігу характерні нахил корпусу вперед, висока амплітуда рухів рук для підтримання рівноваги та додання імпульсу, упор на передню частину стопи під час торкання землі, без подальшого повного перекату, збільшена частота кроків і потужне відштовхування ногами (рис. 1.4).

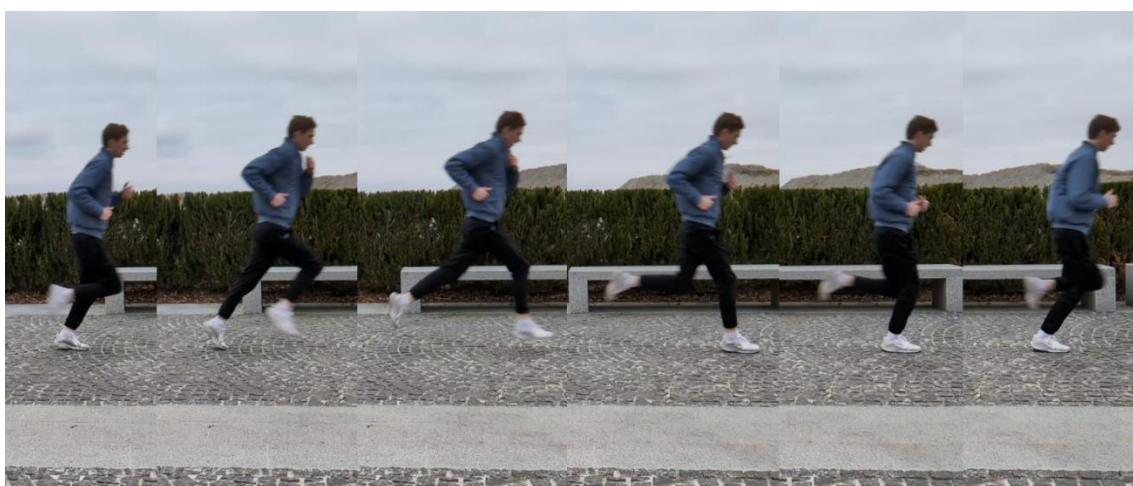


Рисунок 1.4 – Механіка звичайного бігу

Біг підтюпцем – це вид бігу, під час якого тіло переміщується за допомогою чергування рухів ніг із періодичними фазами польоту, але з меншою інтенсивністю та навантаженням на суглоби. Для такого бігу характерні мінімальний нахил корпусу вперед, помірна амплітуда рухів рук для підтримання ритму, плавний переكات стопи з упором на її передню або середню частину, знижена частота і довжина кроків. Біг підтюпцем відрізняється від звичайного бігу більш м'яким відштовхуванням ногами (рис. 1.5).

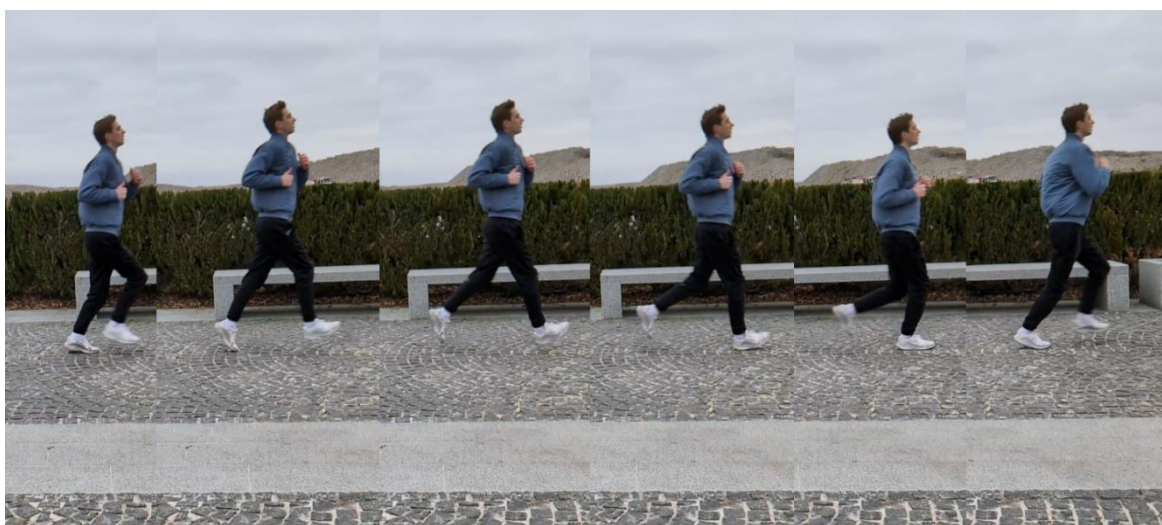


Рисунок 1.5 – Механіка бігу підтюпцем

Проведено порівняння видів локомції (див. табл. 1.1).

Таблиця 1.1 – Порівняння видів локомції

	Ходьба	Біг підтюпцем	Звичайний біг
Циклічність	Присутня	Присутня	Присутня
Довжина кроку	Невелика	Помірна	Велика
Висота кроку	Невелика	Велика	Помірна
Торкання ступи	Повне з перекатом	Повне з перекатом	Упор на передню частину стопи, без перекату
Тривалість торкання	Помірна	Тривала	Нетривала
Частота кроків	Помірна	Висока	Невисока

Кінець таблиці 1.1

	Ходьба	Біг підтюпцем	Звичайний біг
Положення корпусу	Перпендикулярне	Під невеликим нахилом	Під нахилом
Амплітуда руки	Невелика	Помірна	Висока

Вищеописані види локомоції мають низку виразних характеристик, які є основою для їхньої ефективної класифікації.

1.3 Збір даних

Визначившись з властивостями ходьби, звичайного бігу та бігу підтюпцем, можна перейти до вибору ознак для подальшої класифікації. Однак насамперед необхідно визначити джерело даних, на яке можна покладатися. Більшість сучасних смартфонів оснащені вбудованими сенсорами – акселерометром і гіроскопом. Ці сенсори дають змогу вимірювати лінійне прискорення та кутове прискорення, відповідно. Нижче наведено таблицю із застосовністю цих сенсорів (див. табл. 1.2).

Таблиця 1.2 – Ефективне використання сенсорів для отримання властивостей

	Аксилірометр	Гіроскоп
Циклічність	Так	Ні
Довжина кроку	Так	Ні
Висота кроку	Так	Ні
Торкання ступи	Так	Ні
Тривалість торкання	Так	Ні
Частота кроків	Так	Ні
Положення корпусу	Ні	Ні
Амплітуда руки	Ні	Так

Коли користувач робить крок, його телефон за допомогою вбудованих сенсорів відстежує як лінійне, так і кутове прискорення. Ефективність роботи цих сенсорів багато в чому залежить від розташування телефону на тілі користувача. Розміщення пристрою може значно впливати на точність вимірювань та інтерпретацію даних.

Наприклад, якщо телефон знаходиться в кишені штанів, відстеження лінійного прискорення принесе більш корисні дані, ніж кутове прискорення. Це пояснюється тим, що під час підйому й опускання ноги пристрій реєструє характерні зміни лінійного прискорення. Такий тип рухів створює регулярний ритм, що відображається у вигляді послідовності піків і западин у даних сенсора (рис. 1.6).

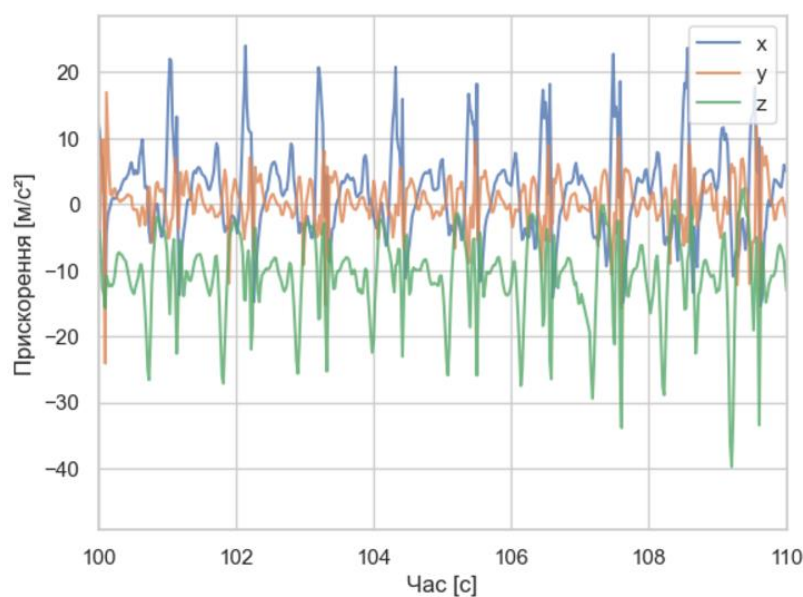


Рисунок 1.6 – Графік лінійного прискорення під час ходьби за осями X, Y і Z

Інший приклад спирається на результати біомеханічних досліджень, які демонструють, що рухи рук і ніг тісно пов'язані та синхронізовані. Коли людина робить крок, її рука робить характерний помах, що відбивається в динаміці руху стопи. Цей взаємозв'язок між рухом руки і ноги дає змогу, аналізуючи рухи руки, визначити момент, коли відбувається крок. Особливо корисним інструментом у цьому випадку стає гіроскоп. Під час періодичного обертання руки, що виникає

під час ходьби, гіроскоп фіксує яскраво виражений синусоїдальний патерн у своєму сигналі. Цей патерн не тільки легко піддається візуалізації, а й дає можливість ідентифікувати моменти, що відповідають крокам, шляхом аналізу піків сигналу гіроскопа [4], [5].

У цьому випадку ми зосереджуємося на ситуації, коли телефон розміщений у кишені штанів. Це положення пристрою є одним з найпоширеніших серед користувачів і дозволяє максимально ефективно використовувати акселерометр.

1.4 Перетворення даних

Розглядаючи дані про прискорення, які подано окремо для кожної осі X, Y і Z (рис. 1.6), можна помітити, що аналіз трьох різних значень одночасно може бути не тільки незручним, а й витратним за ресурсами. Особливо це стає очевидним, коли потрібно проводити обробку даних у реальному часі або робити порівняння між різними моментами часу. Для спрощення та оптимізації подальшого аналізу, ми можемо використовувати формулу Евклідова норма [6] (1.1):

$$r_{mag} = \sqrt{x^2 + y^2 + z^2}, \quad (1.1)$$

де r_{mag} – загальна величина прискорення з гравітацією;

x – прискорення за координатою X;

y – прискорення за координатою Y;

z – прискорення за координатою Z.

Після перетворення даних у загальну величину прискорення, аналіз графіка став значно простішим і зручнішим (рис. 1.7).

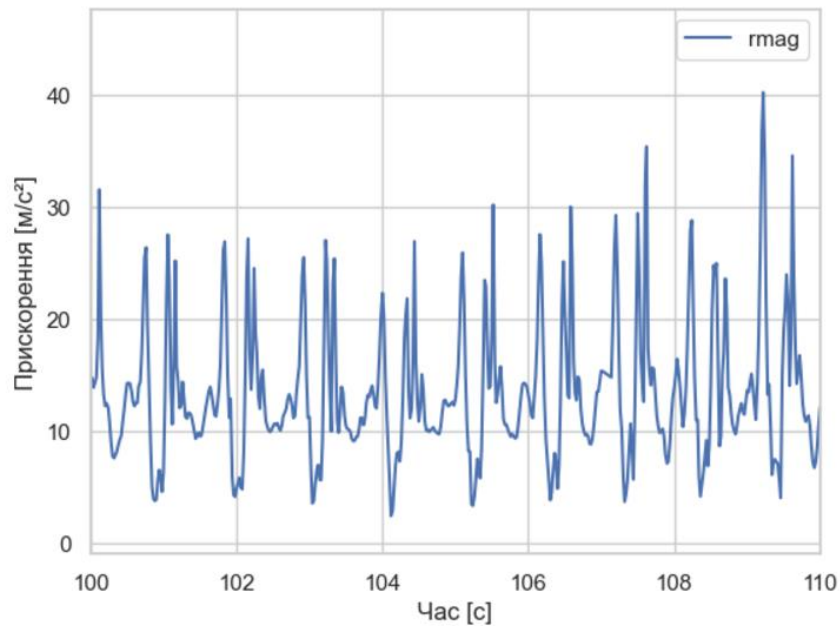


Рисунок 1.7 – Графік з загальною величиною лінійного прискорення

Проаналізувавши графік, можна помітити, що його значення зміщені приблизно на 10 за прискоренням, що відповідає прискоренню сили тяжіння. Для коректного аналізу потрібно усунути вплив гравітації, що дозволить отримати більш точні дані. Для цього слід виконати ще одне перетворення (1.2):

$$mag = rmag - \frac{\sum_{i=0}^n rmag_i}{n}, \quad (1.2)$$

де mag – загальна величина прискорення;

$rmag$ – загальна величина прискорення з гравітацією;

n – розмір масиву.

Результат усунення впливу гравітації можна побачити на графіку нижче (рис. 1.8).

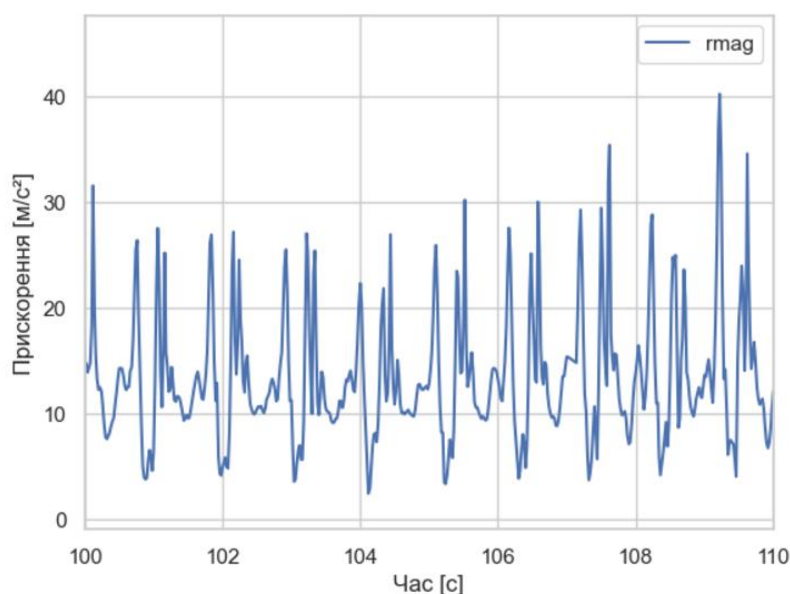


Рисунок 1.8 – Графік з загальною величиною лінійного прискорення без гравітації

1.5 Ознаки кроків

Після виконання останнього перетворення даних ми отримуємо графік, на якому чітко видно регулярні інтервали. Ці інтервали відображають повторювані цикли руху ніг. Якщо вибрати один із таких інтервалів і збільшити його масштаб, стане очевидно, що він являє собою цикл ноги, що включає послідовний рух спочатку однієї, а потім іншої ноги (рис. 1.9).

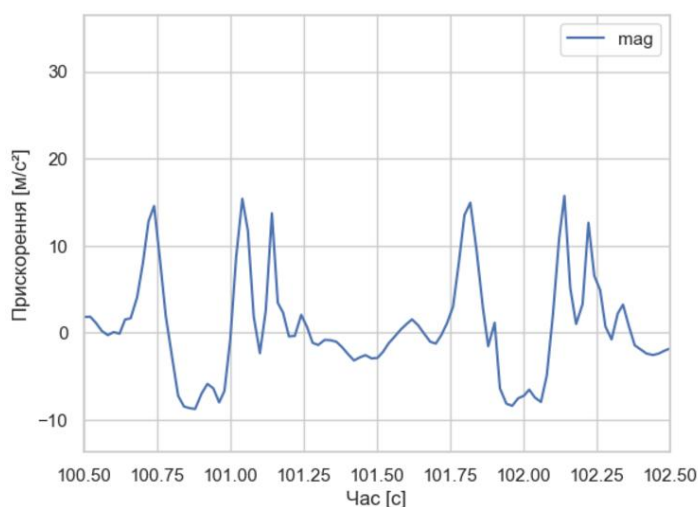


Рисунок 1.9 – Графік циклу ноги

Аналізуючи отриманий графік, можна виділити низку характерних ознак, типових для циклу. Ці особливості допомагають ідентифікувати окремі етапи руху і створити чітку картину циклу ноги. До таких ознак належать:

- найглибша западина;
- дві глибини в найглибшій западині;
- різкий підйом із найглибшої западини;
- найвищий пік після глибокої западини;
- полого довга западина;
- нечіткі дані між найвищим піком і пологою довгою западиною.

Для уточнення взаємозв'язку між сигналами акселерометра і рухами ніг було проведено практичний експеримент. Під час експерименту здійснювалася пряма трансляція даних з акселерометра на екран, що дало змогу в реальному часі спостерігати зміни сигналу під час кожного кроку. Імперицичним шляхом вдалося виявити закономірності, що пов'язують певні ділянки графіка з фазами кроку.

На підставі спостережень було складено таблицю, що демонструє відповідність між етапами руху ноги та характеристиками сигналу акселерометра (див. табл. 1.3).

Таблиця 1.3 – Відповідність між етапами руху ноги та характеристиками сигналу акселерометра

Рух ноги	Характеристика сигналу
Перенесення провідної ноги (з телефоном у кишені) із заднього положення в переднє, іншими словами – крок	Полога довга западина
Удар ноги об землю	Найглибша западина
Удар ноги об землю	Найвищий пік після глибокої западини
Крок другою ногою	Нечіткі дані між найвищим піком і пологою довгою западиною

Після того як було встановлено взаємозв'язок між сигналами акселерометра та кроками під час ходьби, наступним важливим етапом стає визначення характерних ознак для різних типів бігу – звичайного та підтюпцем.

Для цього необхідно провести детальний аналіз графіків, отриманих під час запису сигналів акселерометра під час звичайного бігу та бігу підтюпцем (рис. 1.10).

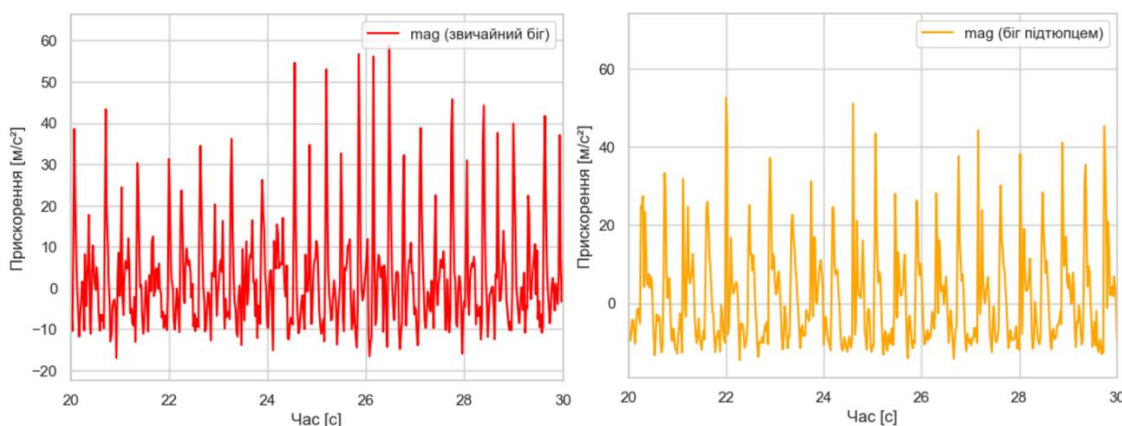


Рисунок 1.10 – Графіки звичайного бігу і бігу підтюпцем

На основі аналізу даних для звичайного бігу і бігу підтюпцем можна зробити висновок, що не всі ознаки, раніше виявлені на графіку для ходьби, можна застосувати до цих видів активності. Наприклад, як:

- дві глибини в найглибшій западині – деякі з найглибших западин мають 3 або більше глибин, а тому ця ознака не характерна для двох видів бігу;
- полого довга западина – ця ознака теж не виявлена.

Крім ознак, які виявилися непридатними для класифікації бігу, було виявлено й загальні ознаки, властиві всім трьом видам локомоції:

- найглибша западина;
- різкий підйом із найглибшої западини;
- найвищий пік після глибокої западини.

Ці загальні ознаки стануть основою для дальшої класифікації. Однак використання цих даних у їхньому вихідному вигляді не дасть достатньої

інформативності для ефективної класифікації кроків. Тому, для досягнення найкращих результатів, дані потрібно привести до фінального вигляду.

Фінальний вигляд ознак кроків буде таким (рис. 1.11):

- прискорення найвищого піку;
- інтервал часу між найнижчою западиною і найвищим піком (1.3);
- інтервал часу між двома повними циклами (1.4).

$$vpi = vt - pt, \quad (1.3)$$

де vpi – інтервал часу між найглибшою западиною і найвищим піком;

vt – час найглибшої западини;

pt – час найвищого піку.

$$pti = pt_{i-1} - pt_i, \quad (1.4)$$

де vpi – інтервал часу між двома повними циклами;

pt – час найвищого піку.

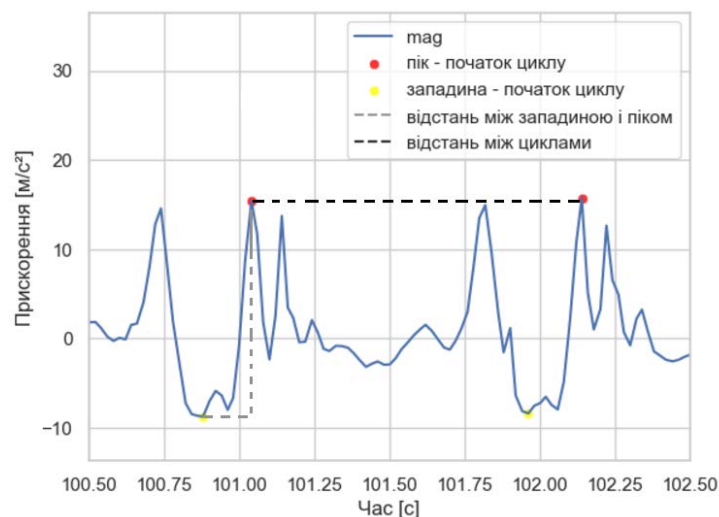


Рисунок 1.11 – Графік двох повних циклів та їх ознаки

У цьому розділі було детально розглянуто характеристики трьох основних видів локомоції: ходьби, бігу підтюпцем і звичайного бігу. Завдяки проведеному аналізу вдалося виділити ключові ознаки, які дадуть змогу класифікувати їх.

2 ВИБІР МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ

2.1 Методи фільтрації даних

Вхідні дані акселірометра потребують фільтра, який буде відповідати таким вимогам:

- легковажність – фільтр повинен швидко обробляти дані, встигати за їх надходженням у реальному часі та не створювати затримок у роботі системи. Це особливо важливо для підтримання ефективності та плавності роботи в умовах постійного потоку інформації;
- пропускання низькочастотних сигналів – у наступних етапах ми будемо очищати дані від близько нульових частот, тому ці сигнали не мають великого значення для подальшого аналізу;
- придушення високочастотного шуму – Для точного аналізу даних критично важливо зменшити вплив шумів, які спотворюють піки та западини сигналу. Очищення сигналу від таких перешкод суттєво підвищує точність визначення ключових характеристик, покращуючи якість подальшої обробки й інтерпретації.

Для задоволення зазначених критеріїв оптимальним вибором є низькочастотний фільтр. Він не лише повністю відповідає усім вимогам, але й вирізняється простою реалізацією, що у сукупності робить його ідеальним кандидатом для цієї задачі.

2.2 Метод класифікації даних

Для вибору класифікатора для проблеми розпізнавання ходьби, звичайного бігу та бігу підтюпцем, було проведено серію експериментів, щоб знайти найбільш підходящий метод, з огляду на вимоги системи і характеристики даних.

Після аналізу результатів експериментів було встановлено, що метод К найближчих сусідів (KNN) продемонстрував найкращий баланс між простотою

реалізації, точністю і швидкістю роботи. Цей класифікатор показав високу точність під час ідентифікації ходьби, а також гідні результати для бігу та бігу підтюпцем.

Ключовою перевагою KNN у цьому завданні стало те, що він ефективно працює з часовими інтервалами даних, де система класифікує не одиничну ознаку, а цілий набір ознак, отриманих за певний період. Такий підхід дає змогу пробачати не найвищу точність методу для звичайного бігу та бігу підтюпцем (рис. 2.1). Але в кінці загальна точність класифікації досягає високого рівня для всіх локомоцій, що робить цей метод особливо придатним для практичного застосування.

precision	
Ходьба	1.00
Біг підтюпцем	0.81
Звичайний біг	0.79
accuracy	
macro avg	0.87
weighted avg	0.84

Рисунок 2.1 – Результати тестових даних

Незважаючи на те, що існують більш складні методи, їхнє використання вимагає значних ресурсів і часу на навчання, що може бути надлишковим для розв'язання цього завдання. Таким чином, метод KNN став оптимальним вибором, що задовольняє всі поставлені критерії.

Для поліпшення результатів було налаштовано параметр числа сусідів K . Для цього було використано метод кроссвалідації, що дало змогу об'єктивно оцінити вплив різних значень K на точність моделі.

Під час цього процесу було враховано кілька важливих аспектів. По-перше, занадто мале значення буде недостатнім для найкращого результату. По-

друге, занадто велике K впливає не тільки на точність, а й на швидкість роботи алгоритму. Для великих значень K час класифікації збільшується, оскільки алгоритм має обробити більше сусідів для прийняття рішення. В умовах проекту це може стати критичним фактором. Таким чином, у процесі налаштування ми дотримувалися балансу між надмірно малими та надмірно великими значеннями K .

У разі потреби система може бути додатково адаптована до нових даних K за допомогою кросвалідації або іншими методами оптимізації. Таким чином, ми досягли не тільки високої якості класифікації, а й гнучкості, що дає змогу враховувати фактори, які змінюються, під час подальшої роботи системи.

2.3 Вибір мови для написання тестової моделі

Перед написанням застосунка, потрібно написати тестову модель класифікатора, у зручному для цього формі, вибором мови для цього став Python. Вибір Python для розробки класифікатора руху, який має розрізняти такі стани, як ходьба, звичайний біг і біг підтюпцем, обґрунтований кількома факторами, зокрема доступними інструментами та зручністю використання мови. Python є однією з найпопулярніших мов програмування в сфері машинного навчання завдяки широкому спектру бібліотек та потужних інструментів, що значно полегшують розробку таких моделей.

Для задачі класифікації руху Python пропонує велику кількість бібліотек, серед яких можна виділити такі:

- NumPy – для роботи з багатовимірними масивами та матрицями, що є основою для обробки даних;
- Pandas – для роботи з таблицями даних;
- Seaborn – для навчальні графіків;
- Scikit-learn – для застосування різних алгоритмів машинного навчання.

Завдяки простоті використання Python і величезній кількості ресурсів для вирішення задач машинного навчання, ця мова є ідеальним вибором для нашого випадку.

2.4 Вибір середовища розробки для написання тестової моделі

Завдання потребує середовища для розробки тестової моделі та перетворення даних. Серед найпопулярніших рішень – PyCharm та Jupyter Notebook.

PyCharm – це потужна IDE, яка забезпечує автодоповнення, дебагер, інтеграцію з системами контролю версій і підтримку плагінів. Вона ідеально підходить для великих, структурованих проєктів, де важливо підтримувати порядок у коді та файлах.

Jupyter Notebook, своєю чергою, орієнтований на інтерактивність і зручність для експериментів. Він ідеально підходить для аналізу даних, візуалізації та тестування моделей. Можливість виконувати код поетапно (блоками) значно спрощує експерименти. Проте управління великими проєктами в Jupyter обмежене через брак засобів для організації файлів.

Для нашого завдання Jupyter Notebook є оптимальним вибором завдяки його простоті та фокусу на інтерактивній роботі з даними. PyCharm, хоч і має потужний функціонал, у цьому випадку буде надлишковим.

2.5 Мови Kotlin та Java: особливості та порівняння

В розробці нативних Android-додатків зазвичай використовуються дві основні мови – Java та Kotlin. Обидві мови мають можливість компіляції написаного коду у ByteCode, який потім виконується на віртуальній машині Java. Саме ця особливість дозволяє добре інтегрувати ці мови між собою [7].

Java, як одна з найстаріших мов для Android-розробки, відома своєю надійністю та зрозумілим синтаксисом для тих, хто знайомий із C-подібними

мовами. Вона пропонує потужну підтримку об'єктно-орієнтованого програмування і демонструє високу продуктивність у багатьох задачах. Однак через явне оголошення типів та і великої кількості на Java може бути тривалішою та більш громіздкою.

Коли ж мова заходить про Kotlin, його основними перевагами стають сучасність і практичність. Kotlin створювався як мова, яка враховує реальні потреби розробників. Він так само підтримує об'єктно-орієнтоване програмування, але додатково дає можливість писати код функціональному стилі, наприклад: Jetpack Compose, Coroutines, Ktor – ці бібліотеки використовують функціональний стиль програмування, що останнім часом стає більш популярною парадигмою програмування, через свої очевидні переваги: як легкість у налагодженні, так і безпеку через імутабельність класів. Крім того, Kotlin має інтуїтивно зрозумілий синтаксис, дозволяє скоротити кількість коду і, найголовніше, забезпечує захист від NullPointerException, який є однією з найбільш поширених помилок у Java.

Загалом, обираючи між цими двома мовами, я схиляюся до Kotlin. Його лаконічність і зручність дозволяють швидше досягати бажаного результату, а сучасні функції та вбудовані механізми безпеки роблять розробку більш приємною і продуктивною. Java, безумовно, залишається потужним інструментом, але її громіздкість і відсутність деяких сучасних можливостей роблять її менш привабливою для нових проєктів.

2.6 Мови Kotlin та Groovy для Gradle: особливості та порівняння

Gradle є основним інструментом збірки Android застосунків, а мови Kotlin і Groovy – це вибір розробників для написання конфігураційних скриптів.

Groovy використовує динамічну типізацію, що дає більше гнучкості, але робить складнішим виявлення помилок під час написання. Виправлення помилок часто потребує часу, оскільки вони проявляються вже під час виконання скриптів.

Kotlin забезпечує неявну статичну типізацію. Kotlin використовує DSL підхід, що допомагає писати більш зрозумілі та структуровані конфігураційні файли, також є забезпечення виявлення помилок на етапі написання коду [8].

Треба зауважити, що написання плагінів для градла на Groovy відбувається найчастіше на Java, яка не є найкращим вибором у порівнянні Java/Kotlin. Kotlin, своєю чергою, дає можливість написання плагінів на самому Kotlin, що покращує інтеграцію плагінів із системою збірки.

Тому вибір очевидний – Kotlin.

2.7 Вибір бібліотеки для зберігання даних

Застосунок буде зберігати кроки та їх тип активності, а для цього потрібно обрати рішення для зберігання даних локально. Room і Realm – це популярні рішення для роботи з базами даних в Android-додатках. Обидва інструменти дають змогу зберігати дані локально, але мають відмінності в архітектурі, функціональності та продуктивності. Детальніше див табл 2.1 [9]:

Таблиця 2.1 – Порівняння Room і Realm

	Room	Realm
Тип бази даних	SQLite	NoSQL
Простота використання	Вимагає знання SQL	Простий інтерфейс для роботи з об'єктами
Продуктивність	Менша для простих операцій	Вища для простих і середніх операцій
Міграція даних	Потрібні сценарії міграції	Автоматична міграція даних
Підтримка	Підтримка Google	Не має підтримки
Кросплатформенність	Ні	Так

Оскільки застосунок у нас нативний, і Room перебуває в просторі Android Jetpack, що означає, що Google рекомендує його використовувати, також бібліотека пропонує кращу масштабованість та інтеграцію з іншими компонентами Android, що робить її оптимальним вибором для цього проєкту. Крім того, підтримка Realm для Android більше не оновлюється, що може призвести до проблем із сумісністю та безпекою в майбутньому. З огляду на ці фактори, вибір падає на Room як більш надійне і сучасне рішення.

2.8 Вибір бібліотеки для впровадження залежностей (DI)

У процесі розробки JetStep виникне велика кількість залежностей між різними компонентами. Це може призвести до ускладнення ініціалізації класів, адже кожен клас може залежати від інших класів або ресурсів, що потребує великої кількості коду для налаштування та ініціалізації. Саме тому виникає необхідність у використанні підходів, які дозволяють полегшити управління залежностями. Одним із таких підходів є патерн Впровадження Залежностей (DI).

DI – це патерн проєктування, що спрямований на спрощення процесу ініціалізації класів шляхом передачі необхідних залежностей зовні замість того, щоб клас самостійно їх створював. Це дозволяє знизити зв'язність компонентів, підвищити їх модульність та зручність тестування. В Android розробці особливо популярні наступні бібліотеки для реалізації впровадження залежностей (див табл 2.2) [10]:

Таблиця 2.2 – Порівняння Dagger 2, Dagger Hilt, Koin

Бібліотека	Переваги	Недоліки
Dagger 2	Висока ефективність та оптимізація коду Підтримка Java та Kotlin Модульність і легкість підтримки коду.	Велика кількість шаблонного коду, повільна компіляція, складність у вивченні.

Кінець таблиці 2.2

Бібліотека	Переваги	Недоліки
Dagger Hilt	Краща інтеграція з Android, менше шаблонного коду, простота вивчення.	Велика кількість шаблонного коду, повільна компіляція, складність у вивченні, менша масштабованість ніж у Dagger 2.
Koin	Невелика кількість шаблонного коду, швидкість розробки, легкість у вивченні, добре інтегрується з Kotlin, мінімальний вплив на швидкість компіляції.	Відсутність компіляційної безпеки, можливі помилки під час виконання, гірша ефективність порівняно з Dagger 2, складніше налагодження програми.

Для мого додатку я обрав Koin як систему впровадження залежностей. Цей підхід є найбільш зручним серед тих, що роздивляються, оскільки Koin мінімізує кількість шаблонного коду, що дозволяє значно пришвидшити розробку та знизити складність налаштування залежностей. Крім того, ефективність, яку пропонує Koin, цілком достатня для потреб мого застосунку, і його інтеграція з Kotlin робить його найкращім вибором для моїх вимог.

2.9 Рішення щодо отримання кроків

Головна вимога програми, це класифікація кроків, тому важливим етапом буде вибір джерела кроків. Для цього в нас є кілька варіантів, використання:

- Google Fit SDK;
- Fit Bit SDK;
- нативний сенсор.

Google Fit SDK – це бібліотека від Google для роботи з даними здоров'я та фітнес-активностями. Серед переваг – висока точність підрахунку кроків та можливість отримання даних пакетами за певні часові інтервали, що дуже зручно для класифікаторів. Основні недоліки: обов'язкова авторизація, складність ініціалізації, а також важливий аспект – припинення підтримки SDK, оскільки Google планує закрити API 30 червня 2025 року [11] .

Fit Bit SDK – це також бібліотека від Google для роботи з даними здоров'я та фітнес активностями, схожа на Google Fit SDK, але має не таку високу точність розпізнавання кроків як вона. Головний недолік, цього рішення – авторизація, вона імплементована дуже не зручно, також обов'язково потрібно отримати офіційний дозвіл на використання SDK [12].

Нативний сенсор – це вбудований функціонал Android для збору даних про фізичну активність. Головні переваги: проста інтеграція і мінімальні залежності. Однак цей підхід має недоліки у вигляді незручного механізму отримання даних та меншої точності розпізнавання кроків.

Зваживши всі переваги й недоліки, вибір припав на нативний сенсор, це легке рішення, яке відповідає вимогам проєкту.

3 РОЗРОБКА ПРОГРАМИ

3.1 Аналіз вимог до застосування

3.1.1 Концепція

Необхідно розробити додаток, що дає можливість користувачам відстежувати їх кроки так види локомоції.

Додаток допомагає користувачеві краще розуміти його фізичні активності за день.

3.1.2 Функціональні вимоги до системи

Користувач повинен мати можливість відстежувати кількість кроків за день.

Користувач повинен мати можливість відстежувати вид локомоції.

Користувач повинен мати можливість бачити розподіл локомоцій в порівнянні з іншими.

Користувач повинен мати можливість додавати кроки та їх класифікацію вручну.

Система повинна зчитувати сенсорні дані коли застосування у фокусі.

Система повинна зчитувати сенсорні дані коли застосування у фоні.

Система повинна зчитувати сенсорні дані коли телефон заблокований.

Система повинна зчитувати сенсорні дані коли застосування вбите.

Система повинна давати можливість зберігати дані кроків.

Система повинна давати можливість класифікувати кроки на основі даних акселерометра.

Система повинна давати можливість класифікувати кроки на три види локомоції: ходьба, звичайний біг, біг підтюпцем.

Система повинна давати можливість автоматично оновлювати інформацію о кроках не рідше ніж раз на хвилину, якщо кроки були зроблені

Система повинна запитувати дозвіл на зчитування активності та відображення повідомлень застосунку у шторці телефону.

Система повинна оновлювати лічильник кроків щодня.

Вихідний код системи має бути захищений шляхом обфускації.

Система повинна унеможливлювати доступ до внутрішньої пам'яті застосунку для перегляду чи маніпуляцій.

3.1.3 Специфікація прецеденту «Відстеження інформації о локомоції»

У таблиці 3.1 наводиться специфікація прецеденту «Відстеження інформації о локомоції».

Таблиця 3.1 – Специфікація прецеденту «Відстеження інформації о локомоції»

Основний виконавець	Користувач
Зацікавлені особи	Додаток
Передумова	Користувач дає дозволи на зчитування активності та відображення повідомлень застосунку у шторці телефону
Постумова	Користувач успішно отримав оновлену інформацію про локомоцію
Основний сценарій	A1. Користувач відкриває додаток. A2. Користувач: a. залишає у фокусі; b. переводить додаток у фон; c. блокує телефон; d. вбиває додаток. A3. Система починає зчитувати кроки та дані аксілерометра. A4. Система піддає дані трансформації, фільтрації та класифікації, по закінченню хвилини зчитування. A5. Користувач бачить інформацію про кроки та їхній вид локомоції.

Кінець таблиці 3.1

Основний виконавець	Користувач
Альтернативні сценарії	В1. Якщо система на основі вхідних даних не має можливості трансформувати, фільтрувати та класифікувати, то система не повертає інформації про кроки, і не зараховує зроблені кроки.

3.1.4 Специфікація прецеденту «Додавання кроків»

У таблиці 3.2 наводиться специфікація прецеденту «Додавання кроків».

Таблиця 3.2 – Специфікація прецеденту «Додавання кроків»

Основний виконавець	Користувач
Зацікавлені особи	Додаток
Передумова	Користувач відкрив додаток
Постумова	Користувач успішно додав кроки та їхній тип, і може побачити як успішно були вони нараховані
Основний сценарій	A1. Користувач перебуває на початковому екрані застосунку. A2. Користувач натискає на кнопку з іконкою редагування. A3. Система показує нижнє модульне вікно для додавання кроків. A4. Користувач вводить кількість кроків. A5. Користувач обирає потрібний тип локомоції шляхом перемикання між ними. A6. Користувач натискає на поле «Виберіть час початку». A7. Система відкриває діалогове вікно з календером. A8. Користувач вибирає день і натискає на кнопку «Ок». A9 Система відкриває діалогове вікно з вибору часу. A10. Користувач вибирає час і натискає на кнопку «Ок». A11. Система ховає діалогове вікно вибору часу. A12. Система ховає діалогове вікно календаря. A13. Система перевіряє, чи всі дані заповнені. A14. Користувач натискає на кнопку «Зберегти». A15. Система зберігає кроки.

Кінець таблиці 3.2

Основний виконавець	Користувач
Основний сценарій	A16. Система відображає збережені кроки на початковому екрані. A17. Користувач бачить інформацію про додані кроки.
Альтернативні сценарії	B1. Після виконання кроку A13, якщо користувач залишив дані неповними, кнопка «Зберегти» залишатиметься неактивною, доки всі поля не будуть заповнені.

3.2 Допоміжний застосунок

Для аналізу предметної області були необхідні дані, на які можна було б спиратися для прийняття рішень. Було розглянуто два можливі підходи:

- використання загальнодоступних даних;
- збір даних самостійно.

Перший варіант мав свої переваги, такі як доступ до великої вибірки даних, яка теоретично могла б забезпечити вищу точність класифікатора. І це може здатися правильним рішенням. Проте цей підхід мав істотний недолік: зібрані дані, не відповідали б специфіці кінцевого тестованого пристрою. Враховуючи, що результати класифікації повинні бути максимально релевантними для конкретного пристрою, спиратися на сторонні дані було визнано недоцільним.

З огляду на це було прийнято рішення на користь другого підходу – збору даних самостійно. Для реалізації цього підходу потрібно було розробити спеціальний застосунок, який би дозволив зібрати дані з акселерометра телефону. Цей підхід забезпечив максимально релевантні дані, адаптовані до конкретного пристрою, що в кінцевому результаті підвищило точність і достовірність роботи класифікатора.

Для початку роботи над допоміжним застосунком потрібно було чітко визначити його функціональність. Основною метою було створити інструмент

для збору даних акселерометра, який би забезпечував зручність використання та точність збору інформації.

Застосунок мав виконувати такі функції:

- початок збору даних акселерометра;
- завершення збору даних;
- перевірка коректності роботи – додати індикатори, які допомагають зрозуміти, чи працює система правильно;
- класифікація зібраних даних;
- очищення даних.

3.2.1 Інтерфейс допоміжного застосунку

Для досягнення цих цілей було створено простий, але функціональний екран (рис. 3.1), що включає такі компоненти:

- кнопка для початку збору даних ходьби;
- кнопка для початку збору даних бігу підтюпцем;
- кнопка для початку збору даних звичайного бігу;
- кнопка для завершення збору даних;
- кнопка для видалення всіх зібраних даних;
- список усіх зібраних даних – дозволяє переглядати вже записані дані;
- відображення останнього значення акселерометра – індикатор перевірки, що дані надходять у реальному часі.



Рисунок 3.1 – Огляд допоміжного застосунка

3.2.2 Модель даних допоміжного застосунку

Щоб організувати збір даних акселерометра та забезпечити їхнє зберігання в базі даних, необхідно було спочатку створити відповідну модель. Ця модель є основою структури даних і визначає, як саме інформація буде зберігатися, оброблятися та використовуватися для подальшого аналізу:

- x – значення прискорення вздовж осі X , яке відображає горизонтальні зміни;
- y – значення прискорення вздовж осі Y , яке відображає вертикальні зміни;

- z – значення прискорення вздовж осі Z , яке відображає зміни глибини;
- `timestamp` – точний час, коли дані були зібрані, необхідний для синхронізації та аналізу часових інтервалів;
- `pace` – тип локомоції, що визначає характер руху (наприклад, ходьба, біг підтюпцем або звичайний біг).

3.2.3 Збирання даних аксілірометра

Для організації збору даних акселерометра в застосунку реалізовано спеціалізований сервіс, який безперервно взаємодіє із сенсором пристрою. Акселерометр працює на частоті 50 Hz, що забезпечує високу деталізацію даних. Отримані дані спершу зберігаються в буфері, після чого групами по 500 записів записуються в базу даних.

Частота 50 Hz обрана як оптимальний баланс між якістю даних і навантаженням на систему [13].

Нижче наведена імплементація:

```
class AccelerometerCollectorService : Service(), SensorEventListener {
    ...
    private var sensorManager: SensorManager? = null
    private var accelerometerRecords =
CopyOnWriteArrayList<mutableListOf<RawAccelerometerRecord>()>()

    override fun onBind(intent: Intent?): IBinder {
        sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
        sensorManager?.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)?.also {
sensor ->
            sensorManager?.registerListener(this, sensor, 20_000) // 50 HZ
        }
        return binder
    }

    override fun onSensorChanged(event: SensorEvent?) {
```

```

    if (event?.sensor?.type == Sensor.TYPE_ACCELEROMETER) {
        accelerometerRecords.add(
            RawAccelerometerRecord(
                x = event.values[0],
                y = event.values[1],
                z = event.values[2],
                recordTakenAt = System.currentTimeMillis(),
                pace = pace
            )
        )
        syncEach500Values()
    }
    ...
}

```

3.3 Архітектура і структура додатка

Застосунок реалізовано відповідно до принципів Чистої Архітектури (Clean Architecture), що передбачає розбиття проекту на модулі згідно з рівнями відповідальності [14].

Переваги модульної структури:

- чітке розділення логіки за рівнями;
- можливість повторного використання коду;
- обмеження доступу до файлів;
- покращення тестованості;
- забезпечення незалежності компонентів.

Модулі застосунку:

а) додаток (app) – виконує роль центрального компонента, що об'єднує всі інші модулі;

б) функції (feature) – містить модулі, присвячені функціональним особливостям застосунку:

1) головний екран (dashboard) – відповідає за інтерфейс користувача (UI) та відображення інформації про кроки;

2) класифікатор локомоції (locomotion-classifier) – забезпечує класифікацію кроків;

в) домен (domain) – відповідає за бізнес-логіку застосунку, зв'язуючи UI з модулем роботи з даними;

г) моделі домену (domain-model) – містить моделі, які використовуються в додатку;

д) тестування домену (domain-test) – допоміжний модуль для створення тестових даних та підробок;

е) дані (data) – забезпечує взаємодію з базами даних. У цьому проєкті відповідає за роботу з локальними базами даних;

ж) ядро (core) – містить модулі, які можуть використовуватися в різних частинах застосунку:

1) інтерфейс користувача (ui) – містить загальні UI-компоненти;

2) загальні елементи (common) – включає утилітарні класи, функції та константи, які не прив'язані до логіки конкретних модулів;

3) дизайн-система (design-system) – відповідає за оформлення застосунку, зокрема кольори та розмірні константи;

4) навігація (navigation) – містить логіку навігації між екранами;

з) логіка збірки (build-logic) – відповідає за конфігурацію збірки проєкту, включаючи плагіни.

Застосунок побудовано таким чином, що кожен модуль може мати залежності від інших модулів, зберігаючи при цьому принципи чіткої ієрархії та незалежності. Структуру модулів і взаємозв'язки між ними наведено на (рис. 3.2).

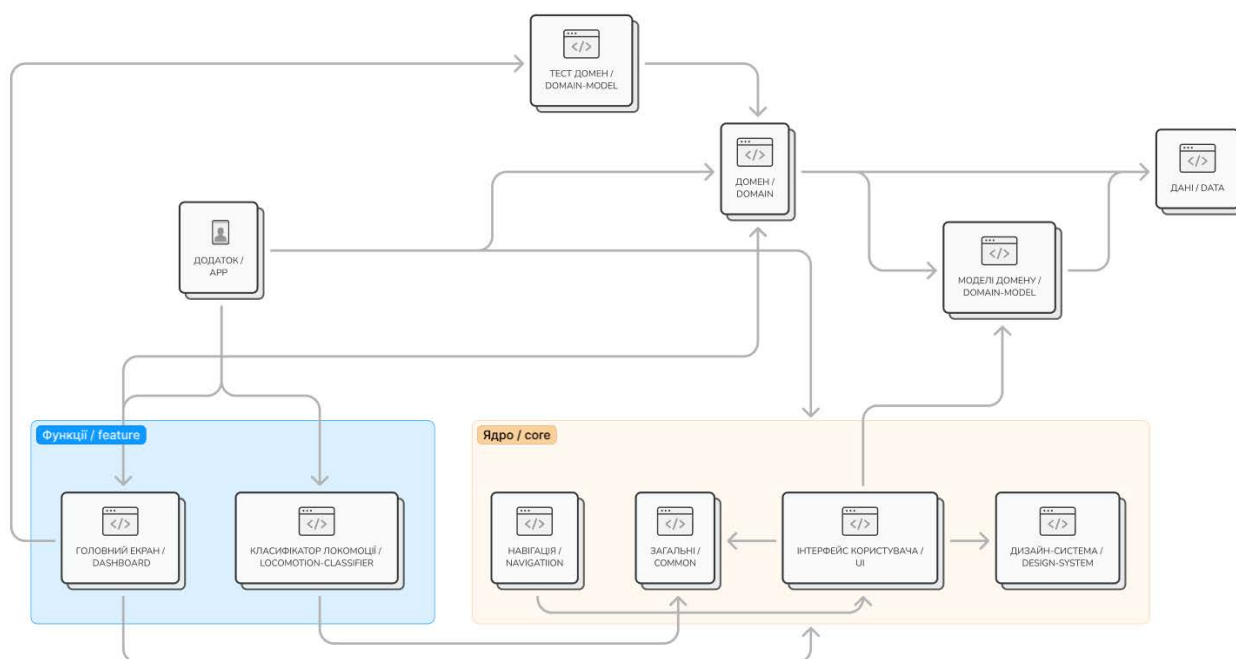


Рисунок 3.2 – Зображення залежностей модулів додатка

3.3 Імплементація патерна проєктування

Додаток імплементує патерн Model-View-Intent (MVI).

MVI – складається з трьох основних компонентів, кожен із яких виконує чітко визначену роль у системі. Цей підхід забезпечує односпрямований потік даних, що полегшує підтримку та розвиток застосунку. Розглянемо компоненти детальніше:

- view – предствлення інтерфейсу користувача;
- view-model – провідний елемент між UI та бізнес-логікою;
- state – тут зберігається стан екрана;
- intent – клас дії, яку потрібно виконати;
- reducer – клас, який відповідає за зміну state, в залежності від отриманого intent;
- usecase – сполучна ланка між доменом і view-model. Виконує обробку intent;
- model – джерело даних.

Механізм функціонування MVI наведено на рисунку 3.3.

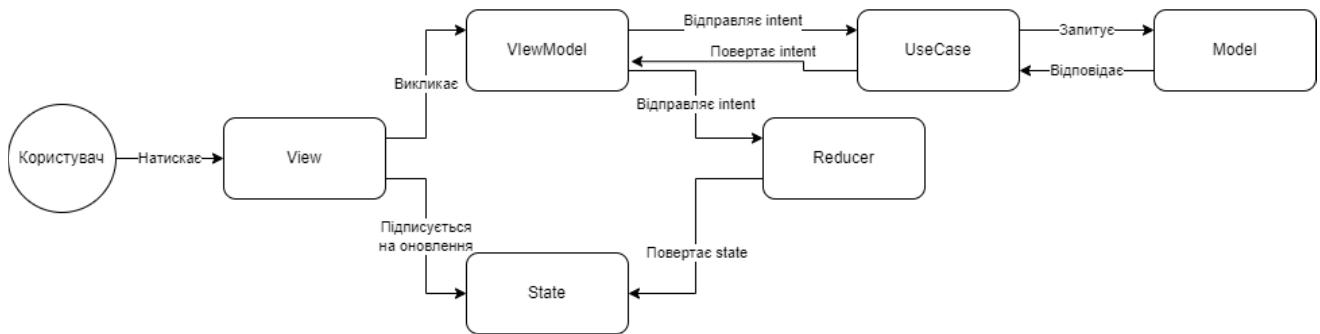


Рисунок 3.3 – Функціонування MVI

Приклад організації файлової системи з використанням патерну проєктування MVI (рис. 3.4). Нижче подано відповідність між реалізованими класами та компонентами MVI:

- DashboardScreen – view;
- DashboardViewModel – view-model;
- DashboardState – state;
- DashboardUiState – state;
- DashboardAction – state;
- DashboardReducer – reducer;
- LoadStepBatchesUseCase – usecase;
- SaveStepBatchManuallyUseCase – usecase.

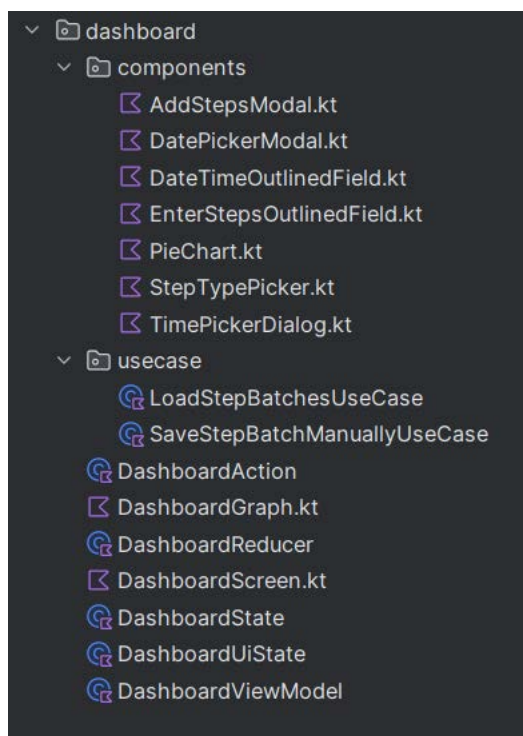


Рисунок 3.4 – Файлової система екрану «Головний»

3.4 Проєктування інтерфейсу

3.4.1 Дизайн система і локалізація

Інтерфейс буде написаний за допомогою Jetpack Compose.

Інтерфейс додатка розроблятиметься за допомогою Jetpack Compose – сучасного інструментарію від Google для створення інтерфейсів користувача в Android-застосунках. Jetpack Compose використовує декларативний підхід, що дозволяє розробникам створювати динамічні та інтерактивні UI з мінімальною кількістю коду порівняно з традиційними методами, такими як XML-маркування [16].

Для впровадження дизайну буде використано Material 3 – сучасну дизайн-систему від Google з відкритим вихідним кодом. Material 3 забезпечує простоту розробки, адаптивність, сучасний вигляд і персоналізацію інтерфейсу для кожного користувача [17].

Щодо локалізації, додаток підтримуватиме дві мови: українську та англійську. Вибір мови здійснюється автоматично через системні налаштування пристрою

3.4.2 Розбір екранів

Додаток містить лише один екран – «Головний», один модальний нижній діалог – «Ручне додавання кроків», а також два діалоги: «Календар» і «Вибір часу» (рис. 3.5).

У наступних розділах буде детально розглянуто кожен екран, їхній функціонал та використані компоненти.

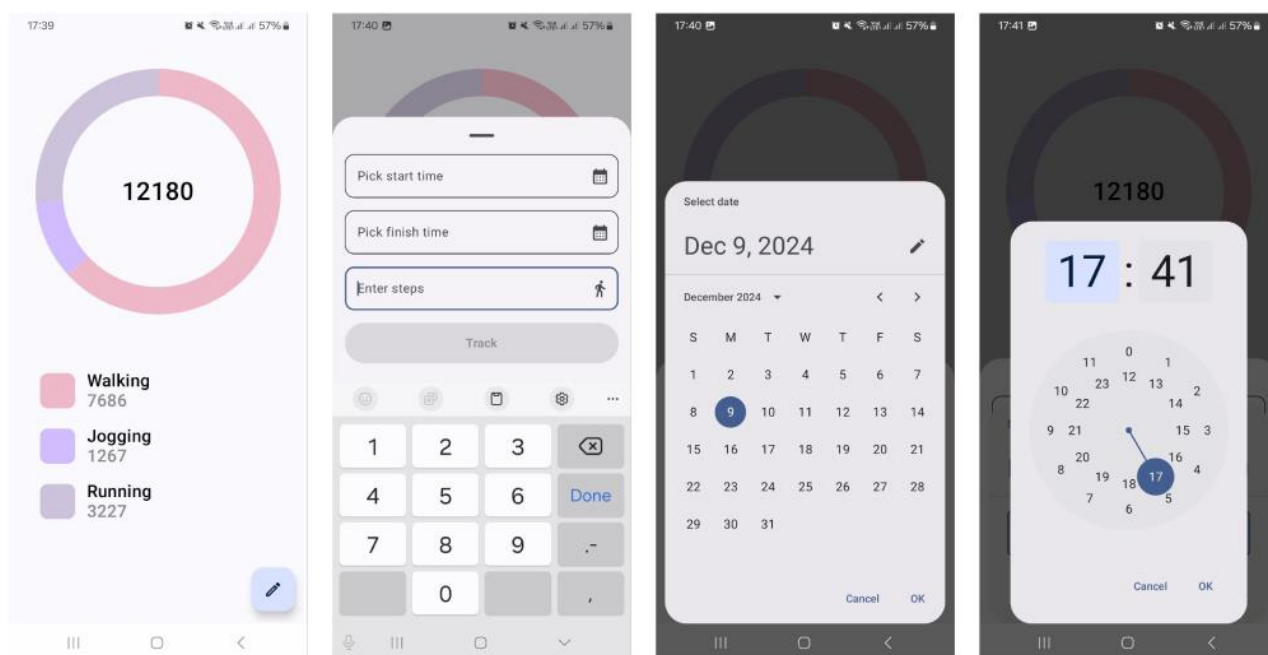


Рисунок 3.5 – Зображення UI компонентів

У наступних частинах буде розбір кожного компонента та їх елементів.

3.4.2.1 Потік екран «Головний»

Екран «Головний» є ключовим елементом структури додатка, виконуючи роль центрального вузла навігації до інших частин інтерфейсу. Основна мета цього екрану – надання користувачу інформації в зручному та зрозумілому форматі (рис. 3.6).



Рисунок 3.6 – Екран «Головний»

На «Головному» екрані, в верхній частині, розташований кільцевий графік, який наочно демонструє розподіл даних за різними категоріями. У центрі цього графіка знаходиться лічильник кроків. Під час ініціалізації цього компонента запускається анімація, яка забезпечує плавний перехід до початкового стану (рис. 3.7).

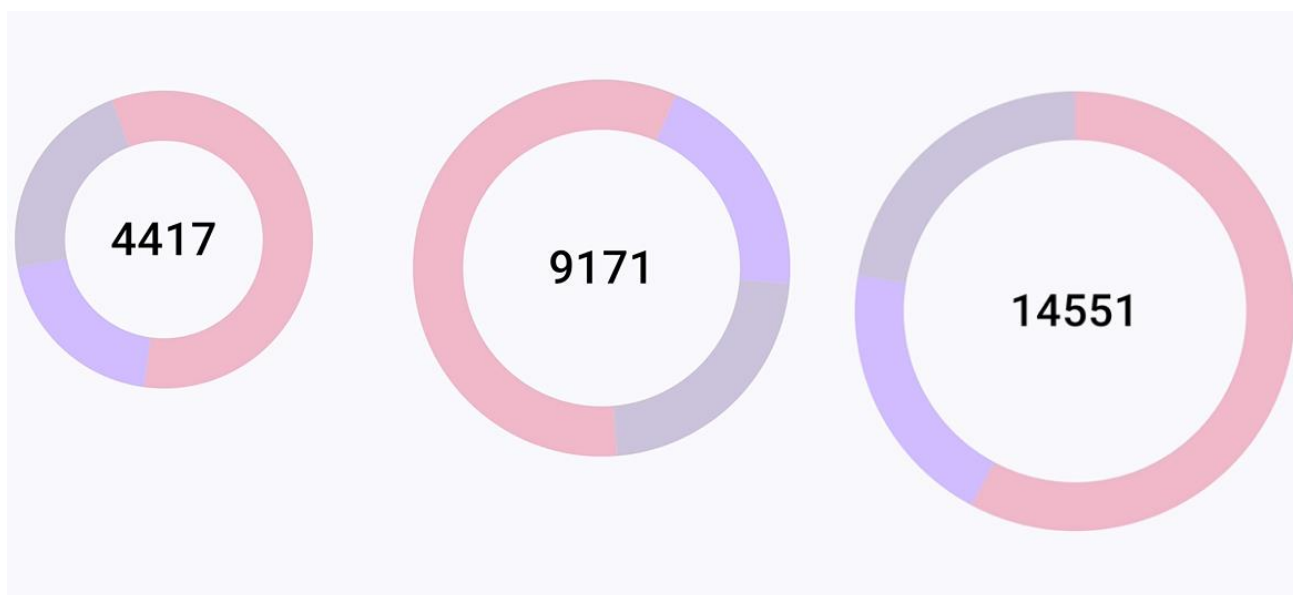


Рисунок 3.7 – Анімація кругового графіку

Під графіком розташований список локомоцій, який містить три типи руху: ходьба, біг підтюпцем та звичайний біг. Для кожної з цих локомоцій вказано кількість зроблених кроків (рис. 3.8).

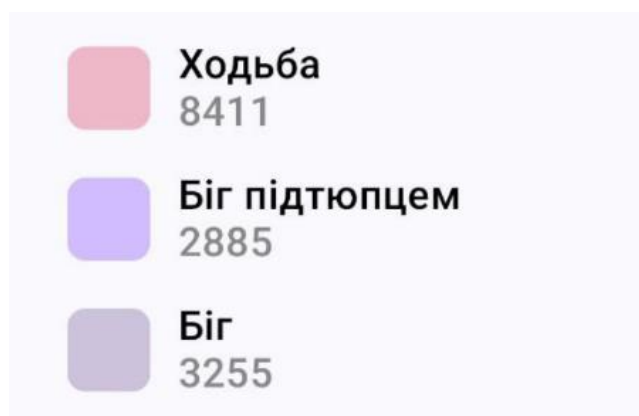


Рисунок 3.8 – Список локомоцій

Оновлення даних цих елементів було реалізовано за допомогою реактивного підходу, що забезпечує автоматичне оновлення інформації. Коли нові дані з'являються в базі даних, наш прослуховувач негайно отримує оновлену інформацію, забезпечуючи безперервну актуальність відображених даних без необхідності додаткових дій з боку користувача.

У правому верхньому куті екрана знаходиться кнопка, яка запускає новий потік під назвою «Додавання кроків вручну». Натискання на цю кнопку відкриває відповідний інтерфейс, дозволяючи користувачу вручну додати кроки (рис. 3.9).



Рисунок 3.9 – Кнопка відкриття потіка «Додавання кроків вручну»

3.4.2.2 Потік «Додавання кроків вручну»

Потік додавання кроків включає такі поля для заповнення:

- час початку відліку кроків;
- час завершення відліку кроків;
- тип кроку;
- кількість кроків.

Потік «Додавання кроків вручну» зображений на рисунку 3.10.

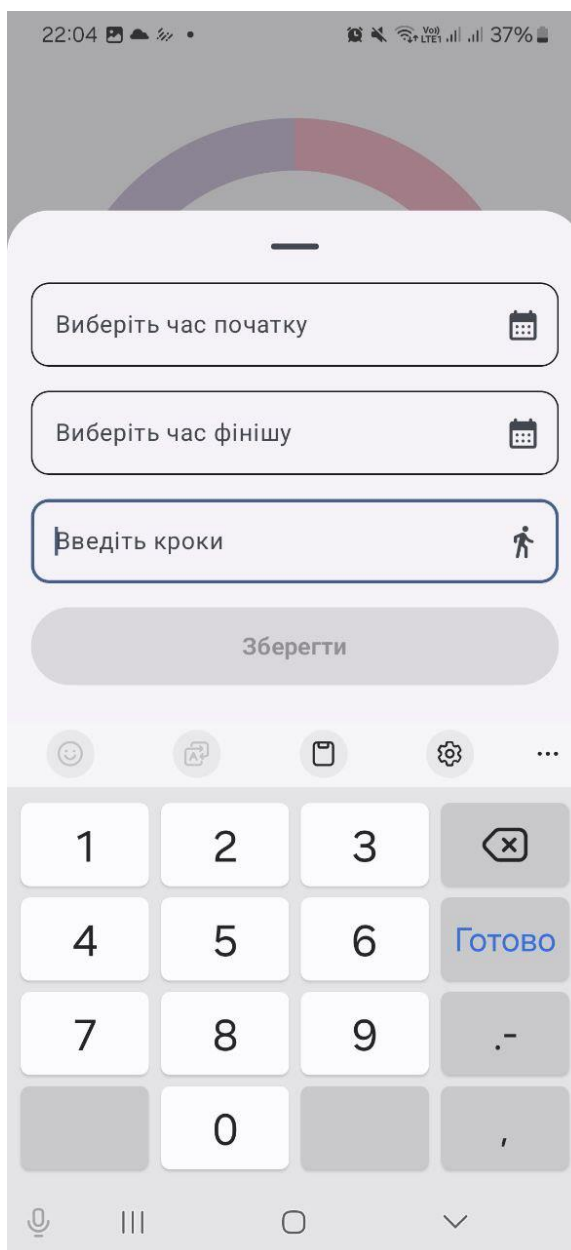


Рисунок 3.10 – Нижній модальний діалог «Додавання кроків вручну»

Кнопка «Зберегти» за замовчуванням неактивна, що візуально позначено приглушеними кольорами. Для її активації необхідно заповнити всі обов’язкові поля.

Тип кроків за замовчуванням встановлений як «Ходьба». Щоб змінити його, користувачу потрібно натиснути на іконку кроків (рис. 3.11). Це ініціює вибір наступного типу локомоції зі списку, причому зміна типу може повторюватися без обмежень.



Рисунок 3.11 – Іконки локомоцій

Щоб розпочати процес встановлення часу початку або завершення відліку кроків, необхідно натиснути на відповідне поле. Після цього з'явиться діалогове вікно з інтегрованим календарем (рис. 3.12), яке дозволяє зручно обрати потрібну дату.

Календар підтримує два способи введення даних:

- вибір дати – користувач має можливість переглядати дати в календарі, зручно перемикаючись між роками та місяцями. Після вибору потрібного періоду дату можна швидко та легко обрати натисканням на відповідний день у календарі;
- введення дати вручну – дозволяє безпосередньо ввести необхідну дату з клавіатури (рис. 3.12).

Для зміни режиму відображення необхідно натиснути на іконку олівця або на іконку календаря (рис. 3.13).

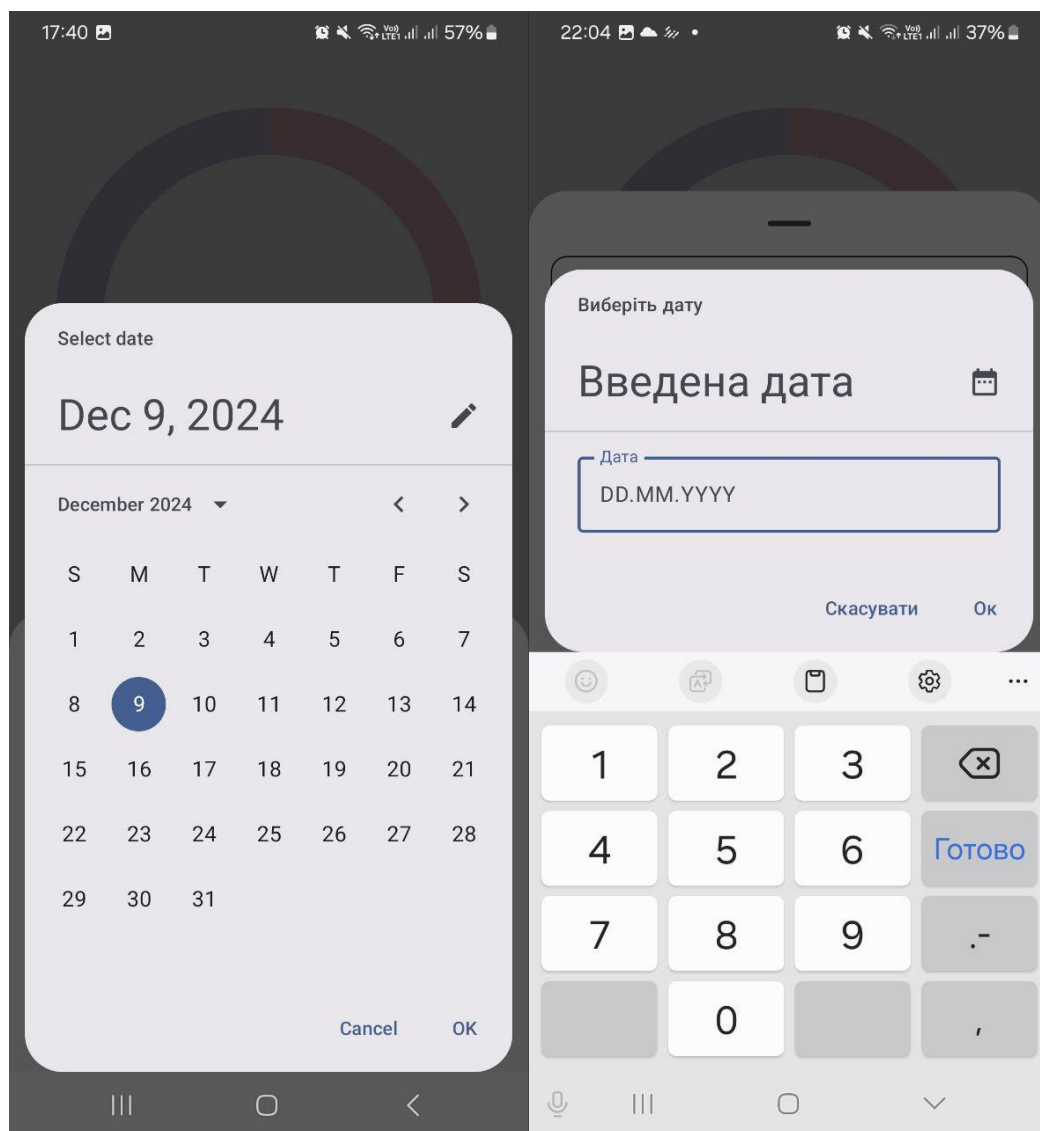


Рисунок 3.12 – Діалогове вікно з календарем (два варіанти)

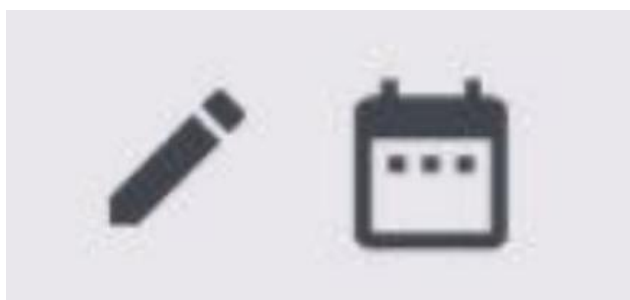


Рисунок 3.13 – Іконки перемикавання режимів календару

Після того як користувач підтвердив вибір дати, відкривається діалогове вікно з вибором часу, вибір відбувається за допомогою натискання на значення на циферблаті, дію повторюють як для годинника, так і для хвилин (рис. 3.14).

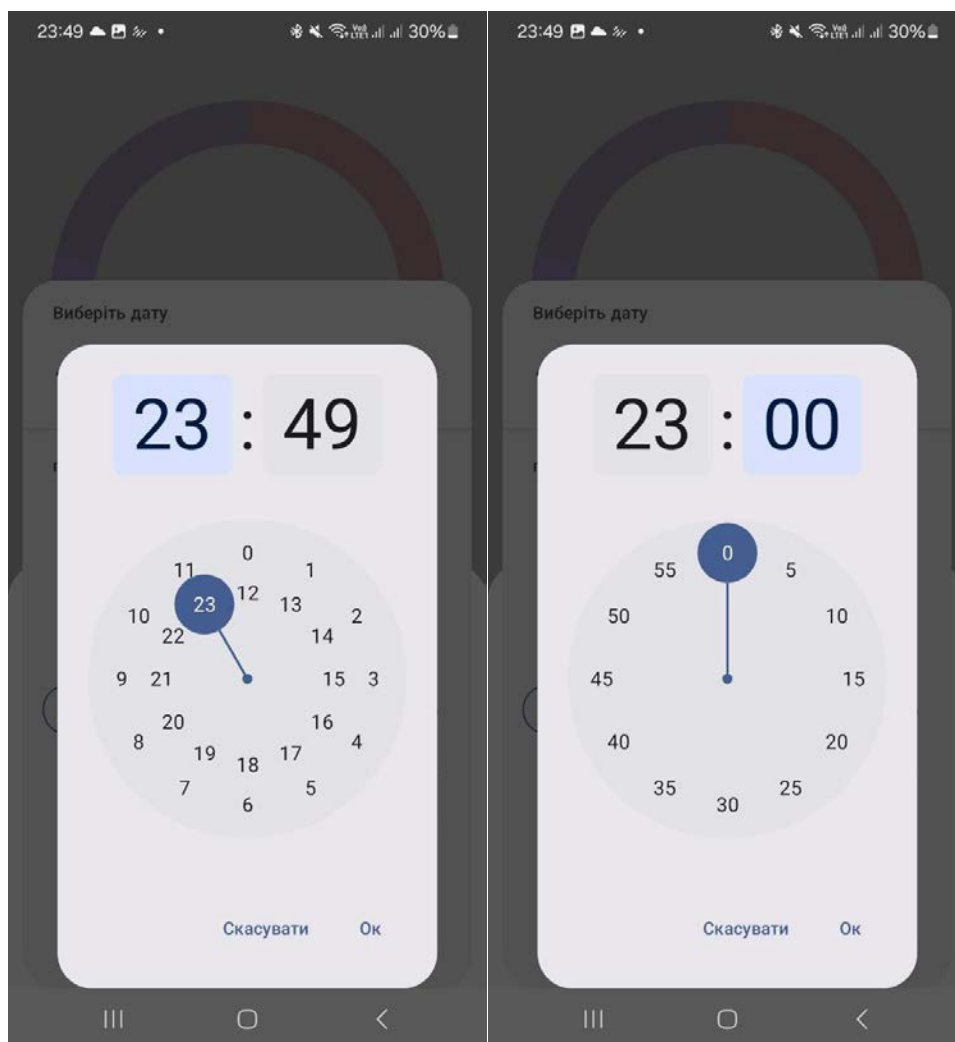


Рисунок 3.14 – Діалогове вікно вибору часу

Після успішного вибору часу, діалогове вікно годинника і календаря закривається. І якщо у нас всі дані заповнені, то кнопка «Зберегти» стає активною (рис. 3.15), після натискання на неї ми зберігаємо дані, і нижнє модальне вікно закривається, після чого реактивним шляхом ми отримуємо оновленні дані.



Рисунок 3.15 – Діалогове вікно вибору часу

3.5. Проєктування даних

Як було зазначено в попередніх розділах, у додатку застосовано підхід Чистої Архітектури, що передбачає поділ на шари. У цьому розділі буде розглянуто шари даних та домену.

3.5.1 Огляд шару data

У додатку зберігається об'єкт `StepBatchLocal`, який представляє сутність для зберігання інформації про певний набір кроків. Цей об'єкт містить такі поля:

- `steps` – кількість кроків за інтервал;
- `startTime` – час початку інтервалу;
- `endTime` – час завершення інтервалу;
- `тип (type)` – тип локомоції.

Цей підхід до зберігання кроків дуже зручний, оскільки він дає змогу не тільки зберігати класифіковані кроки, а й ефективно агрегувати та фільтрувати дані відповідно до заданих правил, наприклад за інтервалом часу.

3.4.1 База даних `SensorDatabase`

Структура таблиці повністю співвідноситься з класом `StepBatchLocal`, нижче наведено код:

```
@Entity(tableName = AccelerometerEntity.TABLE_NAME)
data class AccelerometerEntity(
    @PrimaryKey
    @ColumnInfo(name = COLUMN_TIMESTAMP) val timestamp: Long,
    @ColumnInfo(name = COLUMN_X) val x: Float,
    @ColumnInfo(name = COLUMN_Y) val z: Float,
    @ColumnInfo(name = COLUMN_Z) val y: Float,
    @ColumnInfo(name = COLUMN_PACE) val pace: String,
) {
```

```
internal companion object {

    const val TABLE_NAME = "accelerometer_data"

    const val COLUMN_X = "x"
    const val COLUMN_Y = "y"
    const val COLUMN_Z = "z"
    const val COLUMN_TIMESTAMP = "timestamp"
    const val COLUMN_PACE = "pace"
}
}
```

API таблиці має такий вигляд:

- readInterval(startTime, endTime);
- insert(stepBatches);
- insert(stepBatch).

Нижче представлена ініціалізація бази даних Room, що побудована на основі SQLite:

```
private fun provideSensorDatabase(context: Context): SensorDatabase {
    return Room.databaseBuilder(context, SensorDatabase::class.java,
        SensorDatabase.DATABASE_NAME)
        .build()
}
```

3.5.2 Огляд шару domain

У цьому модулі реалізовано два основні репозиторії: StepRepository (рис. 3.16) та SensorRepository (рис. 3.17).

StepRepository – основна відповідальність цього репозиторію, це надання доступу до даних про зроблені кроки, а також їх збереження у базі даних. Реалізація інтерфейсу репозиторію здійснена в класі StepRepositoryImpl, детальний код якого наведено у додатку А.

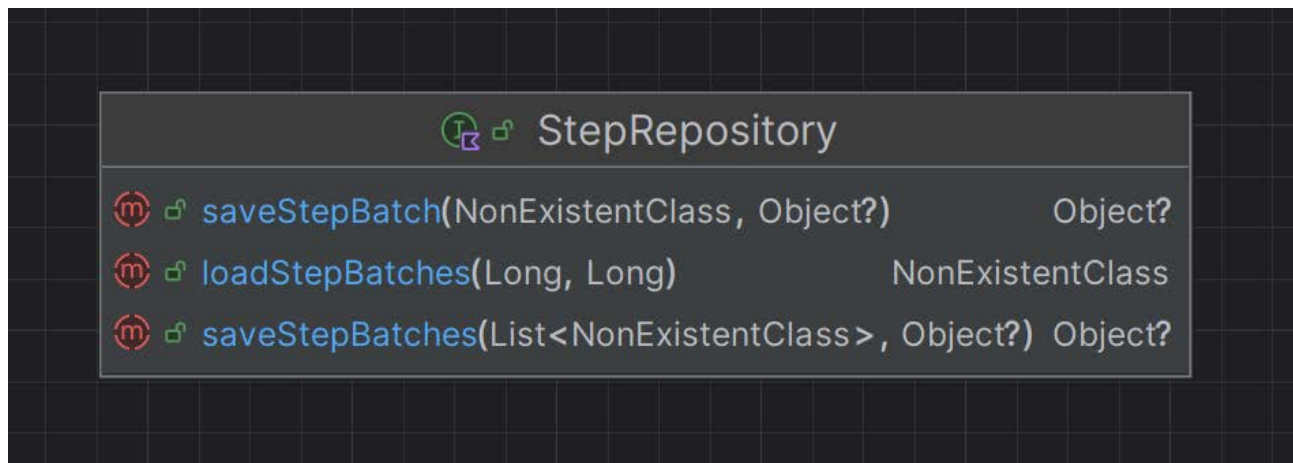


Рисунок 3.16 – API класу StepRepository

SensorRepository – цей репозиторій використовується лише у допоміжному застосунку для збору та зберігання даних акселерометра. Реалізація інтерфейсу представлена в класі SensorRepositoryImpl, також описаному в додатку А.

Перший репозиторій відповідає за надання доступу до зчитування зроблених кроків або збереження кроків.

Реалізація інтерфейсу наведена у додатку А, клас StepRepositoryImpl.

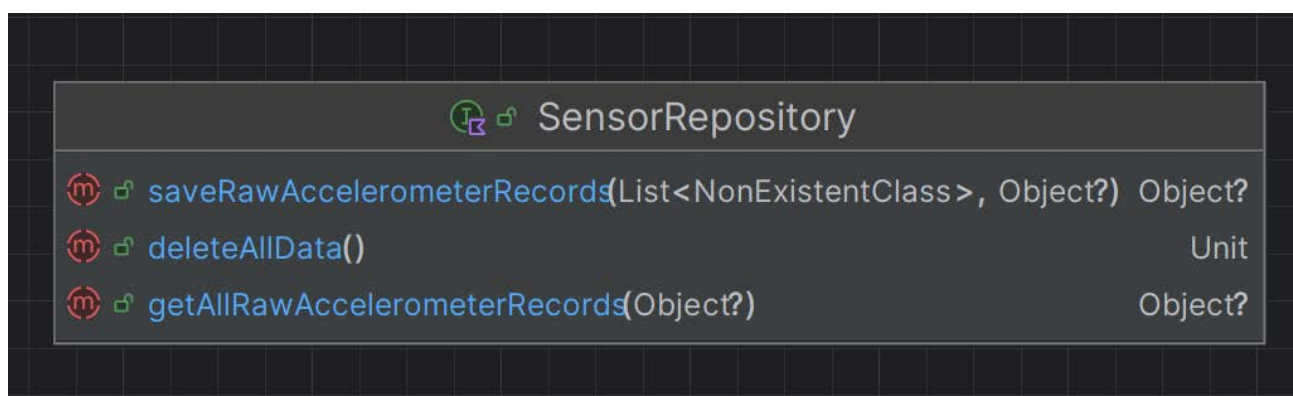


Рисунок 3.17 – API класу SensorRepository

Однією з ключових практик Чистої Архітектури, використаною у цьому проєкті, є розмежування дій класів даних між шарами. Наприклад, у методах saveStepBatch() та loadStepBatches(), які реалізує StepRepositoryImpl,

застосовуються функції-розширення для перетворення даних між різними представленнями:

- `stepBatch.toLocal()` – для перетворення в модель збереження у базі даних;
- `stepBatchLocal.toDomain()` – для перетворення в модель, що використовується у UI шарі.

Якщо в presenter-модулі необхідно використовувати клас, який існує виключно в data-модулі, необхідно створити аналогічний клас, спеціально адаптований для ui-модуля. Такий підхід надзвичайно корисним, особливо коли мова йде про роботу з великими або складними структурами даних, з якими може бути складно працювати безпосередньо в ui-модулі.

У нашому випадку це має ще одну важливу перевагу: підтримка стабільності функцій композицій Jetpack Compose. Ця стабільність є ключовою умовою для коректного відображення та плавної роботи UI.

3.6 Збір даних

Збір даних здійснюється за допомогою сервісу переднього плану, що дозволяє підтримувати безперервний потік інформації навіть у випадках, коли застосунок перебуває у фоновому режимі, телефон заблокований або застосунок примусово закрито.

Для створення такого сервісу необхідно виконати декілька кроків:

- а) реєстрація сервісу – реєстрація сервісу в `AndroidManifest.xml`;
- б) вказати тип сервісу – починаючи з Android 14, потрібно обов’язково зазначати тип сервісу. У нашому випадку, обирається тип «здоров’я» (health) [18];
- в) вказати дозволи – реєстрація дозволів в `AndroidManifest.xml`:
 - 1) `foreground_service` – дозвіл на використання сервісу переднього плану;

2) foreground_service_health – дозвіл на використання сервісу переднього плану з типом «здоров'я»;

3) post_notifications – на відображення повідомлень;

4) activity_recognition – цей дозвіл дозволяє застосунку фіксувати дії користувача;

г) запитати дозволи – запит здійснюється під час запуску застосунку.

Нижче наведений код AndroidManifest.xml (app):

```
<manifest>

    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission
android:name="android.permission.FOREGROUND_SERVICE_HEALTH"/>
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
    <uses-permission android:name="android.permission.ACTIVITY_RECOGNITION"/>

    <application>

        ...

        <service
            android:name=".android.services.FitnessForegroundService"
            android:exported="false"
            android:foregroundServiceType="health" />

    </application>
</manifest>
```

У самому сервісі відбувається реєстрація сенсорів для зчитування двох основних видів даних:

- кроків;
- акселерометра.

Для роботи з акселерометром використовуються ті самі налаштування, які були використані в допоміжному застосунку.

Нижче наведений код надсилання даних класу LocomotionHelper:

```

class FitnessForegroundService : Service(), SensorEventListener {
    ...
    private val locomotionHelper: LocomotionHelper by inject()

    override fun onSensorChanged(event: SensorEvent?) {
        when (event?.sensor?.type) {
            Sensor.TYPE_ACCELEROMETER -> locomotionHelper.feedAccelerometer(
                x = event.values[0],
                y = event.values[1],
                z = event.values[2]
            )

            Sensor.TYPE_STEP_COUNTER -> locomotionHelper.feedSteps(
                aggregatedStepsCount = event.values[0].toLong()
            )
        }
    }
    ...
}

```

3.7 Перетворення даних

3.7.1 Агрегація зібраних даних

Після передачі даних у `LocomotionHelper`, клас виконує функцію обробки та запису отриманих даних. Зібрані дані розподіляються між двома списками:

- акселерометра;
- кроків.

Кожен елемент у цих списках має поле часової позначки (`timestamp`), яке вказує, коли саме він був зафіксований.

Процес агрегації зібраних даних базується на часових інтервалах. Ми обираємо відрізок тривалістю одна хвилина і відфільтровуємо всі зібрані дані, які відповідають цьому інтервалу часу. Це дозволяє створювати структуровані записи, прив'язані до конкретного періоду. Такий підхід є зручним для подальшої аналітики.

Після завершення процесу агрегації створюється клас **StepDetails**, який відіграє ключову роль у подальшому перетворенні та обробці даних.

3.7.2 Перетворення в загальну величину прискорення

У розділі 1.4 зазначалося, що для коректної обробки даних акселерометра необхідно привести значення X, Y та Z до єдиної величини, що дозволить отримати загальну величину прискорення. Після цього необхідно виключити вплив гравітації, яка є постійною складовою вимірювань сенсору акселерометр. Загальна величина прискорення має таку структуру моделі **AccelerometerMagnitude** (рис. 3.18).

Далі наведено код, що реалізує формули (1.1) та (1.2):

```
private val accelerometerMagnitudeRecords: List<AccelerometerMagnitude>
    get() {
        return printDuration {
            val rmags = accelerometerRecords.map { accelerometerRecord ->
                AccelerometerMagnitude(
                    collectedAtMillis = accelerometerRecord.collectedAtMillis,
                    magnitude = sqrt(
                        accelerometerRecord.x * accelerometerRecord.x +
                        accelerometerRecord.y * accelerometerRecord.y +
                        accelerometerRecord.z * accelerometerRecord.z
                    )
                )
            }

            val gravity = rmags.sumOf(AccelerometerMagnitude::magnitude) /
rmags.size

            rmags.map { rmag ->
                rmag.copy(magnitude = rmag.magnitude - gravity)
            }.mapLowPassFilter(alpha = LOW_PASS_FILTER_ALPHA)
        }
    }
```

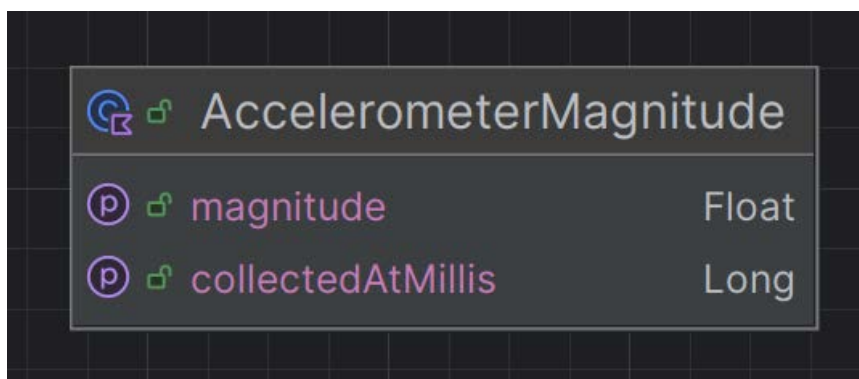


Рисунок 3.18 – Структура класу AccelerometerMagnitude

3.7.3 Фільтрація шумів

Сигнал акселерометра часто супроводжується значним рівнем шуму, що може бути спричинено як зовнішніми факторами, так і особливостями роботи самого сенсора, інтегрованого в телефон. Щоб забезпечити точність отриманих даних та мінімізувати вплив шумів, необхідно застосувати фільтрацію. Для цього була реалізована функція, яка впроваджує фільтр низьких частот, дозволяючи відсіяти небажані високочастотні коливання. Код реалізації цієї функції наведено нижче:

```

fun List<AccelerometerMagnitude>.mapLowPassFilter(alpha: Float):
List<AccelerometerMagnitude> {
    return mapIndexed { index, accelerometerMagnitude ->
        if (index == 0) {
            accelerometerMagnitude
        } else {
            accelerometerMagnitude.copy(
                magnitude = (alpha * get(index - 1).magnitude) + (1 - alpha) *
accelerometerMagnitude.magnitude
            )
        }
    }
}

```

3.7.4 Фільтрація вершин та западин

Після проведення фільтрації шумів на попередньому етапі сигнал акселерометра все ще містить велику кількість точок, що може ускладнювати подальший аналіз і обробку даних. Для оптимізації роботи з сигналом було реалізовано механізм зменшення кількості точок, який базується на відборі ключових елементів – вершин і западин.

Принцип роботи алгоритму полягає в ітераційному аналізі кожної точки сигналу у порівнянні з її сусідніми значеннями. Якщо значення точки є вершиною (вища за обидві сусідні точки) або западиною (нижча за обидві сусідні точки), і при цьому перевищує або дорівнює визначеному мінімальному порогу, ця точка вважається ключовою та зберігається. Усі інші точки, які не відповідають цим критеріям, виключаються з подальшої обробки.

Застосування цього алгоритму дозволило значно скоротити обсяг даних для аналізу, залишаючи лише ті точки, які мають ключову інформацію для характеристики сигналу. Наприклад, початкова кількість точок у сигналі становила 12,053, але після застосування алгоритму вона скоротилася до 1,936 (рис. 3.19).

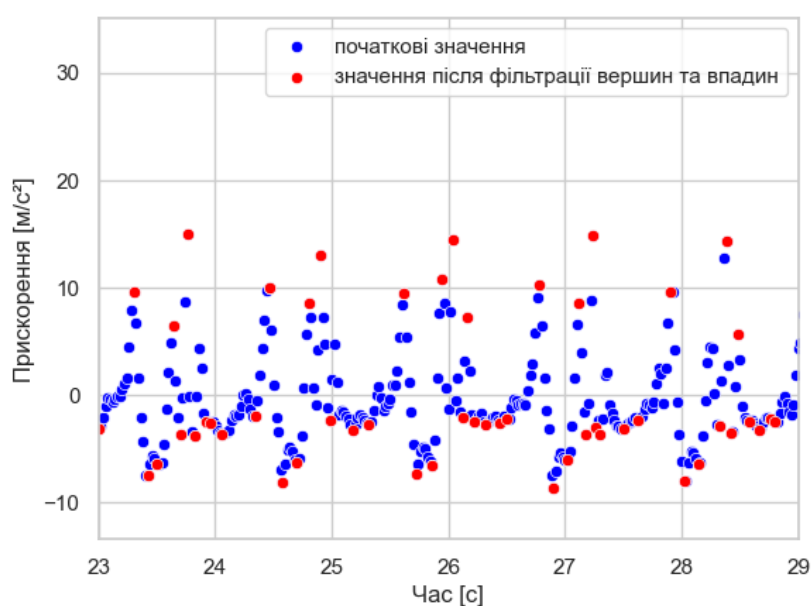


Рисунок 3.19 – Порівняння даних до фільтрації вершин та впадин та після

Нижче наведена імплементація:

```
fun List<AccelerometerMagnitude>.findPeakSimple(
    threshold: Float,
    reversed: Boolean = false,
): List<AccelerometerMagnitude> {
    val filteredValues = mutableListOf<AccelerometerMagnitude>()
    val coef = if (reversed) -1 else 1
    val threshold = threshold * coef
    for (index in range(0, size)) {
        val prev = (getOrNull(index - 1)?.magnitude ?: Float.MIN_VALUE) * coef
        val current = get(index).magnitude * coef
        val next = (getOrNull(index + 1)?.magnitude ?: Float.MIN_VALUE) * coef
        when {
            prev > current && current < next && current <= threshold -> {
                filteredValues.add(get(index))
            }
            else -> continue
        }
    }
    return filteredValues
}
```

3.7.5 Фільтрація локальних вершин та западин

Попередній метод фільтрації за вершинами та западинами, хоча і забезпечують базову обробку даних, але його не достатньо для якісної оцінки сигналу для подальшого визначення ключових точок. Щоб досягти вищого рівня чистоти та точності, було вирішено застосувати вдосконалений підхід – фільтрацію за локальними екстремумами в межах визначених часових інтервалів.

Суть цього методу полягає у поділі сигналу на короткі часові відрізки, у нашому випадку тривалістю по 300 мілісекунд – це значення було визначено експериментально, шляхом перебору різних інтервалів, і показало себе як оптимальне для нашого випадку. У межах кожного такого інтервалу виконується пошук локальних мінімумів (западин) та максимумів (вершин). Точки, які не є

локальними екстремумами, видаляються, оскільки вони не несуть цінної інформації для аналізу.

У результаті застосування цього методу вдалося зменшити кількість точок у сигналі з 1937 до 1566 (рис. 3.20).

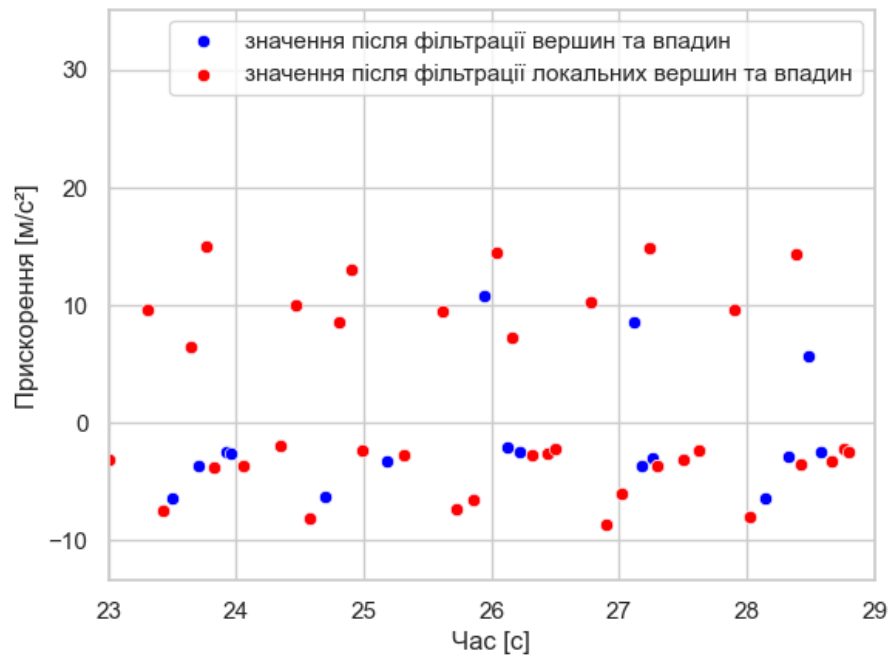


Рисунок 3.20 – Порівняння даних до фільтрації локальних вершин та впадин та після

Реалізація фільтрації локальних вершин та впадин, наведена у додатку А, клас файл FindLocalPeaksAndValleysFilter.kt.

3.7.6 Знаходження істинної вершини та западини

Після застосування всіх попередніх етапів фільтрації, ми можемо визначити справжні вершини та западини, які представляють завершений цикл руху ноги, тобто два кроки. Для цього необхідно виконати наступний алгоритм:

- спочатку береться інтервал у 500 мілісекунд і в ньому шукається мінімум (западина);

- від точки знайденого мінімуму відраховується наступний інтервал у 500 мілісекунд, у якому виконується пошук максимуму (вершини);
- якщо в цьому інтервалі максимум не був знайдений, то знайдений раніше мінімум не вважається валідним і відфільтровується;
- ці операції повторюються ітераційно до завершення аналізу всього сигналу.

Цей підхід базується на аналізі характерного патерну руху під час ходьби, який було визначено емпірично. Застосування цього методу дозволило виокремити 723 точки із 1566 точок сигналу (рис. 3.21).

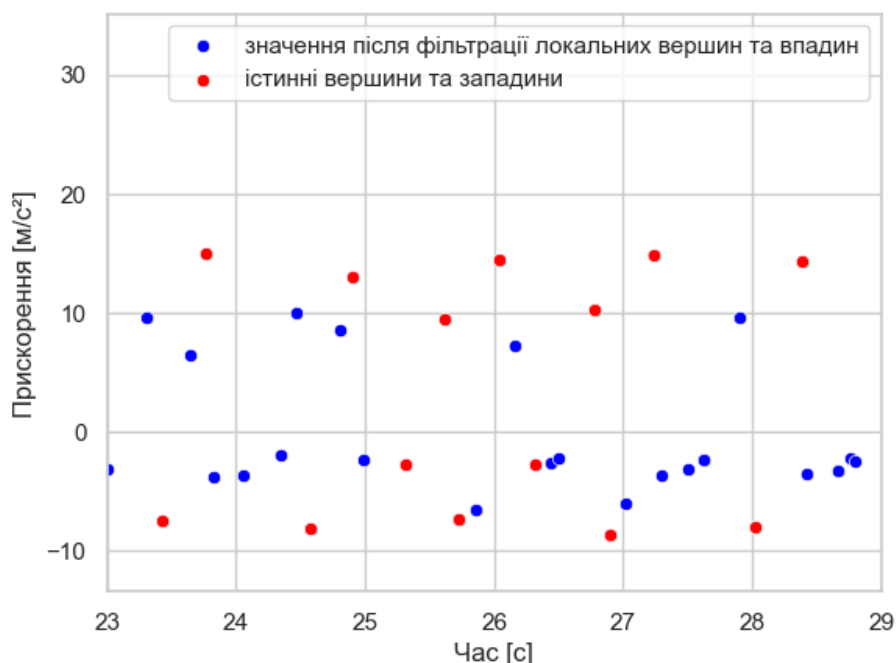


Рисунок 3.21 – Порівняння даних до істинних вершин та впадин та після

Отримані результати переводяться у новий формат даних для подальшого використання. Структуру нового типу даних можна побачити на малюнку 3.22.

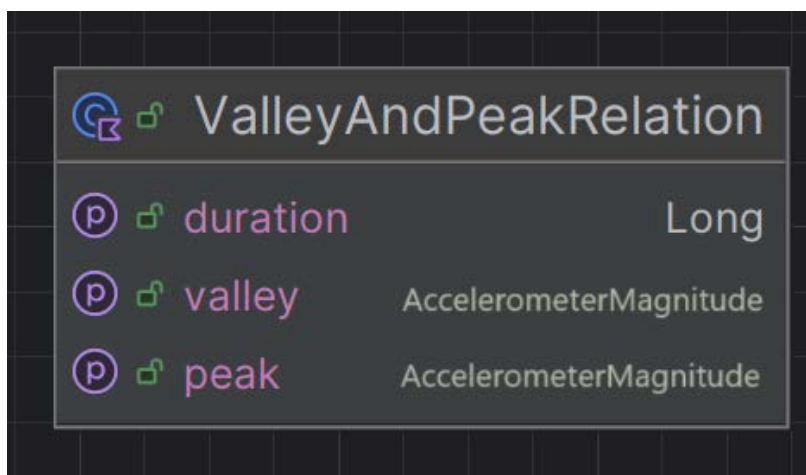


Рисунок 3.22 – Структура одного циклу ноги

3.7.7 Трансформація даних в ознаки

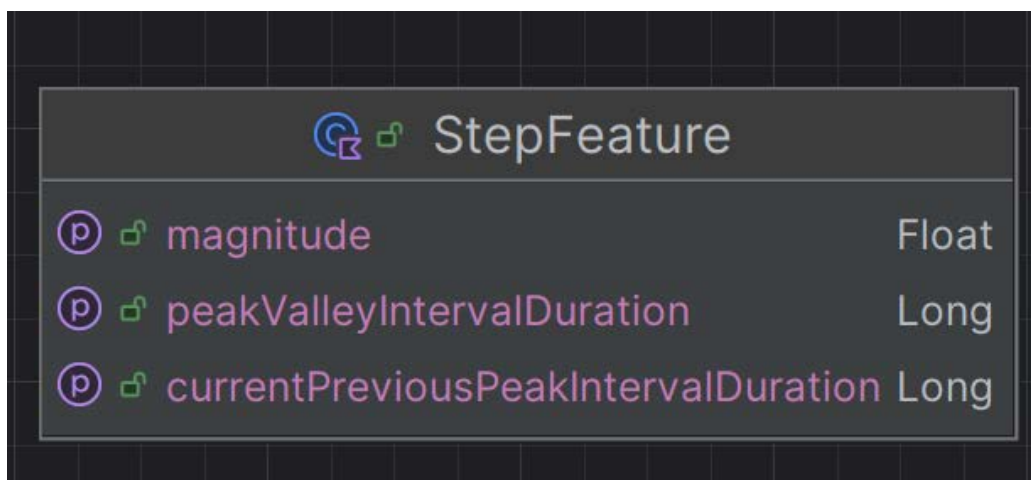
Наступний етап – це трансформація отриманих даних в ознаки. Використовуючи клас «ValleyAndPeakRelation», що містить істинні вершини та западини сигналу, ми можемо обчислити ключові параметри для подальшого аналізу.

Першою ознакою є часова відстань між істинною западиною та її вершиною.

Другою ознакою є часова відстань між двома послідовними вершинами, що відповідає повному циклу руху ноги (два кроки).

Третя ознака – це прискорення у вершині. Під час експериментів було розглянуто можливість використання значень прискорення як у вершині, так і в западині, однак прискорення у вершині виявилось більш стабільним і виразним. Саме його було обрано для подальшого аналізу.

Отримані ознаки утворюють структуру, яку можна побачити на рисунку 3.23. Клас StepFeature дає змогу структурувати зібрані дані акселерометра у формі, зручній для класифікації.



StepFeature	
(p)  magnitude	Float
(p)  peakValleyIntervalDuration	Long
(p)  currentPreviousPeakIntervalDuration	Long

Рисунок 3.23 – Структура одного циклу ноги

Останній етап фільтрації спрямований на уточнення даних для класифікації шляхом знаходження моди часової відстані між циклами кроків. Для обчислення моди беруться часові відстані між вершинами, які були визначені на попередніх етапах обробки даних. Мода дозволяє виділити найбільш характерний інтервал для досліджуваної локомоції.

Наступним кроком є застосування отриманого значення моди з допустимим відхиленням у 100 мілісекунд, як критерій фільтрації. Усі цикли, у яких часові відстані виходять за межі встановленого інтервалу відсіюються. Це дозволяє видалити шумові дані, що могли залишитися після попередніх етапів фільтрації. У результаті цієї процедури формується остаточний список ознак.

3.8 Розпізнавання локомоцій

Для підготовки навчальних даних класифікатора ми виконали обробку та трансформацію даних із попередніх розділів, зберігши їх у форматі CSV. Було створено три окремі файли, кожен з яких відповідає одному з типів локомоції:

- ходьба;
- біг підтюпцем;
- звичайний біг.

Ці файли були завантажені в середовище Jupyter Notebook для подальшої роботи за допомогою бібліотеки Pandas. Наступним етапом стало маркування кожного набору даних. Для цього ми вручну додали колонку для класифікації типу локомоції:

- 0 – для ходьби;
- 1 – біг підтюпцем;
- 2 – звичайний біг.

Після цього всі три таблиці були об'єднані в один спільний DataFrame (структура даних для роботи з табличними даними). Цей крок дозволяє створити єдиний набір даних, який можна використовувати для подальшого навчання моделі класифікації (рис. 3.24).

```
[351]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df_walking = pd.read_csv("phonerresults/walking_results.csv")
df_walking['pace'] = 0
df_jogging = pd.read_csv("phonerresults/jogging_results.csv")
df_jogging['pace'] = 1
df_running = pd.read_csv("phonerresults/running_results.csv")
df_running['pace'] = 2

df = pd.concat([df_walking, df_jogging, df_running], ignore_index=True, sort=False).sample(frac=1).reset_index(drop=True)
df.head(6)
```

	cycle_diff	valley_peak_diff	mag	pace
0	640	400	6.918552	0
1	681	100	30.361101	2
2	639	420	28.870811	2
3	400	160	14.528324	1
4	380	200	53.852090	1
5	419	159	22.228884	1

Рисунок 3.24 – Імпортування даних у Jupyter Notebook

Після цього кроку ми можемо перейти до візуалізації отриманих даних, щоб краще зрозуміти їх розподіл (рис. 3.25).

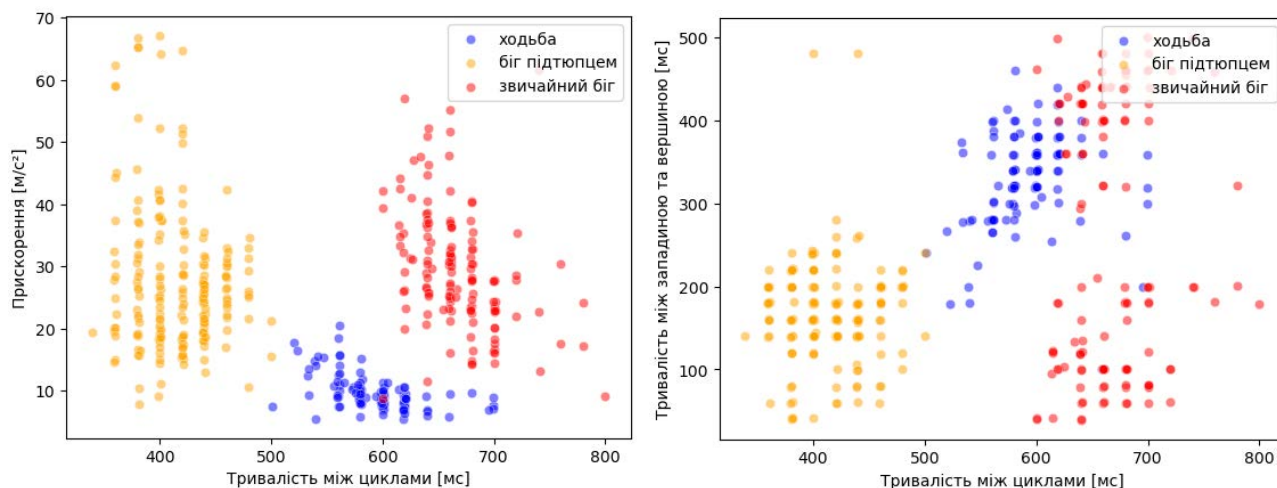


Рисунок 3.25 – Візуалізація ознак

Переходимо до навчання моделі. На першому етапі розподіляємо дані на тренувальні та тестові набори за допомогою функції `train_test_split` з бібліотеки `Scikit-learn`.

Наступним обов'язковим кроком є масштабування даних. Це пояснюється тим, що алгоритм К найближчих сусідів (K-Nearest Neighbors, KNN) базується на розрахунку відстаней між точками даних. Якщо різні ознаки мають неоднакові масштаби, це може призвести до некоректних результатів. Для масштабування використовуємо клас `StandardScaler` із бібліотеки `Scikit-learn`.

Після масштабування даних можна перейти до безпосереднього навчання моделі. Алгоритм KNN реалізовано в класі `KNeighborsClassifier` бібліотеки `Scikit-learn` (рис. 3.26).

```
[377]: from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

X = df.drop('pace', axis=1)
y = df['pace']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

scaler = StandardScaler()
scaled_X_train = scaler.fit_transform(X_train)

model = KNeighborsClassifier(n_neighbors=1)
model.fit(scaled_X_train, y_train)
```

[377]: KNeighborsClassifier

KNeighborsClassifier(n_neighbors=1)

Рисунок 3.26 – Комірка з навчанням моделі

Щоб виконати класифікацію даних, спочатку необхідно масштабувати тестові дані так само, як це було зроблено для тренувального набору. Це забезпечує узгодженість між обома наборами даних і гарантує, що модель отримує дані в однаковому масштабі. Масштабування тестових даних здійснюється за допомогою вже навченого на тренувальних даних об'єкта масштабування `StandardScaler`.

Після масштабування тестових даних можна здійснити класифікацію за допомогою навченої моделі KNN.

Для отримання детального аналізу класифікації слід скористатися функцією `classification_report` з бібліотеки `Scikit-learn`. Ця функція генерує звіт (рис. 3.27).

```
[383]: from sklearn.metrics import confusion_matrix, classification_report

scaled_X_test = scaler.transform(X_test)
y_pred = knn_model.predict(scaled_X_test)
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.96	0.96	0.96	46
1	0.98	0.98	0.98	56
2	0.96	0.96	0.96	50
accuracy			0.97	152
macro avg	0.97	0.97	0.97	152
weighted avg	0.97	0.97	0.97	152

Рисунок 3.27 – Комірка класифікації тестового набору даних

Щоб підвищити точність класифікації, доцільно провести підбір оптимальної кількості сусідів. Це можна здійснити шляхом перебору значень `n_neighbors` конструктора класу `KNeighborsClassifier` (рис. 3.28).

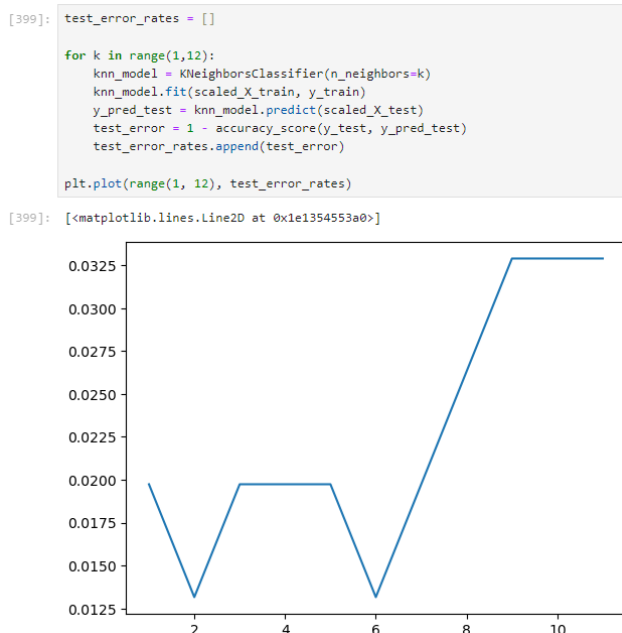


Рисунок 3.28 – Комірка підбору оптимальної кількості сусідів

За результатами перебору видно, що оптимальна кількість сусідів становить 2.

3.9 Інтеграція класифікатора з Android

Для інтеграції навчених моделей було використано бібліотеку Sklearn-onnx. Необхідно експортувати не лише модель KNN, а й StandardScaler, який потрібен для масштабування ознак (рис. 3.29).

```
[245]: from skl2onnx import convert_sklearn
from skl2onnx.common.data_types import FloatTensorType

initial_type = [('feature_input', FloatTensorType([None,3]))]
model_path = 'phonerresults\sklearn_model.onnx'
scaler_path = 'phonerresults\sklearn_scaler.onnx'

# Write the ONNX model to disk
converted_model = convert_sklearn(model, initial_types=initial_type)
with open(model_path, "wb") as f:
    f.write(converted_model.SerializeToString())

converted_model = convert_sklearn(scaler, initial_types=initial_type)
with open(scaler_path, "wb") as f:
    f.write(converted_model.SerializeToString())
```

Рисунок 3.29 – Комірка експорту

Після цього потрібно оптимізувати експортовані файли командами:

```
python -m onnxruntime.tools.convert_onnx_models_to_ort sklearn_model.onnx
```

```
python -m onnxruntime.tools.convert_onnx_models_to_ort sklearn_scaler.onnx
```

Для імпорту файлів, їх потрібно перенести в директорію ресурсів raw (рис. 3.30). В коді прописати ініціалізацію класу OrtSession, код ініціалізації та виклик моделі наведений в додатку А, клас LocomotionClassifierImpl.

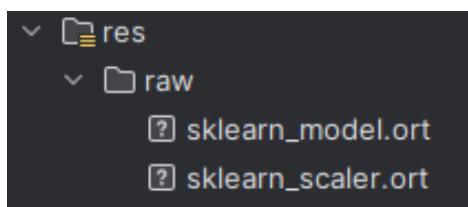


Рисунок 3.30 – Директорія raw

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ

4.1 Створення захищеного APK

Для створення захищеного та оптимізованого APK необхідно активувати використання R8 та вимкнути режим налагодження.

Для цього потрібно встановити значення перемінної `isDebuggable` на `false` у файлі `build.gradle.kts` в модулі додатка в секції конфігурації збірки для типу "випуск".

```
android {  
    buildTypes {  
        getByName("release") {  
            isDebuggable = false  
        }  
    }  
}
```

Для того щоб додати R8 до проекту, необхідно в файлі `build.gradle.kts` додати відповідні налаштування. Це включає в себе активацію мінімізації коду та оптимізації, що дозволяє зменшити розмір APK і ускладнити зворотну інженерію коду.

```
android {  
    buildTypes {  
        getByName("release") {  
            isMinifyEnabled = true  
            isShrinkResources = true  
            proguardFiles(  
                getDefaultProguardFile("proguard-android-optimize.txt"),  
                "proguard-rules.pro"  
            )  
            signingConfig = signingConfigs.named("release").get()  
        }  
    }  
}
```

```
}  
}  
}
```

Для зміни типу збірки з `devDebug` на `prodRelease`, потрібно відкрити варіанти збірки, натиснути на `app` модуль, та обрати `prodRelease` (рис. 4.1).

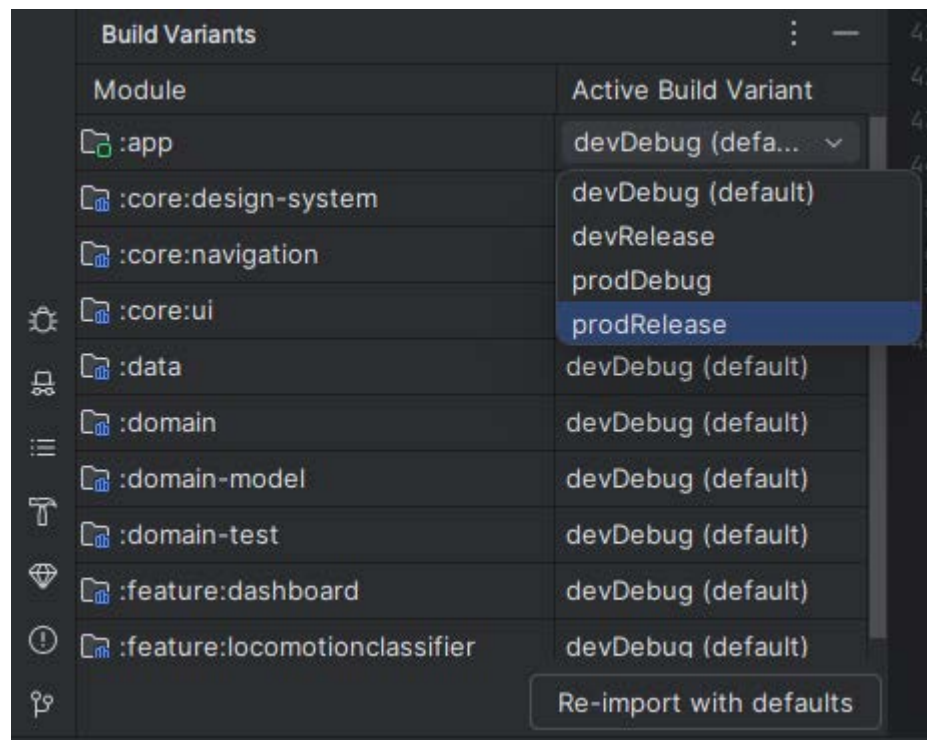


Рисунок 4.1 – Вибір збірки «`prodRelease`»

Для збірки проекту потрібно натиснути на «`Build`» в панелі інструментів, а потім обрати «`Build APK`» (рис. 4.2).

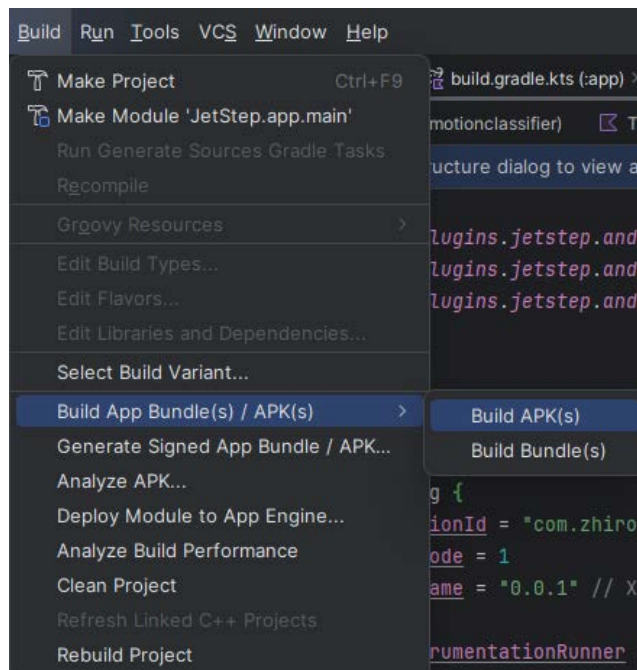


Рисунок 4.2 – Збірка APK

Після завершенню збірки APK, з'явиться повідомлення (рис. 4.3).

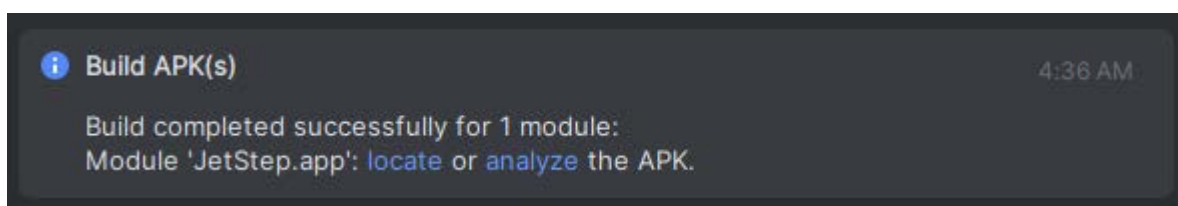


Рисунок 4.3 – Повідомлення о завершенні збірки

4.1.1 Порівняння збірок debDebug та prodRelease

За допомогою Android Studio можна подивитись, чи є доступ до внутрішніх даних програми, або ні. Якщо відкрити APK через будь-який архіватор, то можна отримати доступ до внутрішнього коду програми.

У таблиці 4.1 наводиться порівняння debDebug та prodRelease збірок.

Таблиця 4.1 – Порівняння debDebug та prodRelease збірок

	debDebug	prodRelease
Доступ до внутрішніх даних	Так	Ні
Розмір додатку	85 МБ	75 МБ
Обфускація	Ні	Так
Захищеність	Ні	Так
Швидкість роботи додатку	Прийнятна	Швидко

Як можна помітити, після вимкнення режиму налагодження Android система блокує доступ до внутрішніх файлів додатка. Це забезпечує додатковий рівень від несанкціонованого доступу. Водночас, після застосування R8, розмір APK зменшився на 10 МБ. Також це позитивно вплинуло на швидкість завантаження та ефективність роботи додатку. Крім того, завдяки обфускації, код програми став значно менш читабельним, що ускладнило його аналіз і зворотну інженерію.

4.2 Експлуатація додатка

Щоб розпочати користування додатком, потрібно відкрити програму, натиснувши на ярлик JetStep.

Після відкриття додатка необхідно виконати такі кроки:

- надати дозвіл на відображення повідомлень;
- надати дозвіл на розпізнавання активностей.

Після надання доступів, щоб коректно почати збір даних, потрібно зробити такі дії:

- покласти телефон у кишеню;
- виконувати одну з трьох локомоцій (ходьба, біг підтюпцем або звичайний біг);
- протягом 60 секунд продовжувати виконувати обрану дію.

Дістати телефон із кишені та переглянути кільцеву діаграму з лічильником, де буде відображено кількість нових кроків.

Щоб додати кроки вручну, натисніть на кнопку в правому нижньому кутку з іконкою олівця.

З'явиться нижнє модальне вікно. В ньому треба зробити:

- заповнити поле кількості кроків;
- обрати тип локомції, натисканням на іконку руху;
- почати потік вибору часу, після натиску на поле «Виберіть час початку» або «Виберіть час фінішу», щоб відкрити діалогове вікно з календарем.

У діалоговому вікні обрати потрібну дату. Після вибору дати відкриється діалогове вікно для вибору часу.

Виберіть потрібний час і натисніть ОК.

Система поверне вас до нижнього модального вікна. Тепер можна натиснути на уже активну кнопку «Зберегти».

Якщо кроки додавали на поточний день, вони будуть автоматично додані до загальної кількості кроків за сьогодні.

4.3 Тестування додатка

4.3.1 Тестування сервісу переднього плану

Для перевірки роботи сервісу, необхідно впевнитись, що сервіс працює коректно. Перш за все, слід перевірити, чи відображається відповідне системне повідомлення. Якщо повідомлення відображається, це є першим підтвердженням того, що сервіс активний (рис. 4.4).

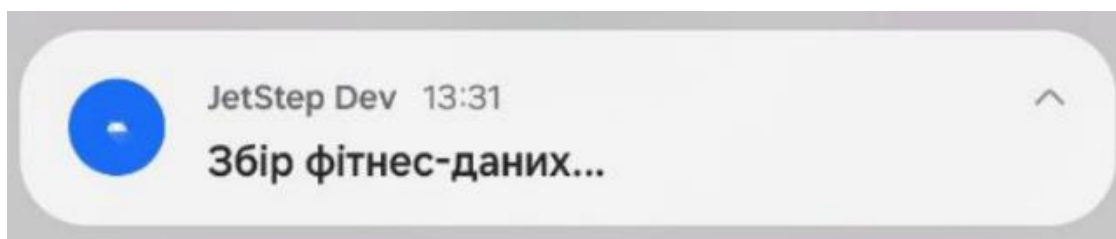


Рисунок 4.4 – Повідомлення про роботу сервіса

Окрім цього, для додаткової впевненості в роботі сервісу, у код було додано спеціальний рядок, який друкується в консолі під час активності сервісу (рис. 4.5). Завдяки цьому, можна відстежити стан роботи сервісу та переконатися, що він функціонує належним чином (див. табл. 4.2).



```
com.zhiro.jetstep.dev I I'm still working!
pid-1513 E attributionTag not
com.zhiro.jetstep.dev I I'm still working!
```

Рисунок 4.5 – Повідомлення в консолі

Таблиця 4.2 – Порівняння тривалості життя фітнес сервісу

	1 хв.	3 хв.	5 хв.	15 хв.	30 хв.	60 хв.
Застосунок відкритий	Так	Так	Так	Так	Так	Так
Застосунок в фоні	Так	Так	Так	Так	Так	Так
Застосунок вбитий	Так	Так	Так	Так	Так	Так
Телефон заблокований	Так	Так	Так	Так	Так	Так
Застосунок вбитий та телефон заблокований	Так	Так	Так	Так	Так	Так

Як можна побачити з табличних даних, програма працює без проблем зі зчитуванням даних у сервісі за різних умов.

4.3.2 Тестування точності зчитування кроків

Для тестування точності зчитування кроків, було створено таблицю для кожного виду локомоції, у кожній таблиці є 3 ітерації, з різним розподілом кроків (див. табл. 4.3-4.5).

Таблиця 4.3 – Таблиця кроків під час ходьби

	5 кроків	30 кроків	100 кроків
Перша ітерація	0	32	104
Друга ітерація	0	29	97
Третя ітерація	0	30	101

Таблиця 4.4 – Таблиця кроків під час бігу підтюпцем

	5 кроків	30 кроків	100 кроків
Перша ітерація	0	27	101
Друга ітерація	0	30	105
Третя ітерація	0	31	100

Таблиця 4.5 – Таблиця кроків під час звичайного бігу

	5 кроків	30 кроків	100 кроків
Перша ітерація	0	0	114
Друга ітерація	0	32	104
Третя ітерація	0	35	107

Як можна побачити, що все перші 5 кроків не були зараховані. Це пов'язано зі специфікою фільтрації кроків, яка базується на аналізі розпізнаної локомоції. В процесі роботи алгоритм потребує певного часу для стабілізації сигналу і коректного визначення типу активності.

Найвищий показник точності розпізнавання було досягнуто для ходьби, що свідчить про відносно простий і передбачуваний характер сигналу. Друге місце за точністю посідає біг підтюпцем, який демонструє трохи більшу складність у розпізнаванні. Найнижчий показник спостерігається для звичайного бігу, що пояснюється більш складною природою сигналу, зумовленою високою інтенсивністю рухів та різкими змінами динаміки.

4.3.2 Тестування точності класифікації кроків

Для тестування точності розпізнавання рухів, ми створили таблицю для кожного виду локомоції, у кожній таблиці є 3 ітерації, з різним розподілом кроків (див. табл. 4.6-4.8).

Таблиця 4.6 – Таблиця розпізнавання руху під час ходьби

	5 кроків	30 кроків	100 кроків
Перша ітерація	–	Біг підтюпцем	Ходьба
Друга ітерація	–	Біг підтюпцем	Ходьба
Третя ітерація	–	Ходьба	Ходьба

Таблиця 4.7 – Таблиця розпізнавання руху під час бігу підтюпцем

	5 кроків	30 кроків	100 кроків
Перша ітерація	–	Біг підтюпцем	Біг підтюпцем
Друга ітерація	–	Біг підтюпцем	Біг підтюпцем
Третя ітерація	–	Звичайний біг	Біг підтюпцем

Таблиця 4.8 – Таблиця розпізнавання руху під час звичайного бігу

	5 кроків	30 кроків	100 кроків
Перша ітерація	–	–	Біг підтюпцем
Друга ітерація	–	Звичайний біг	Звичайний біг
Третя ітерація	–	Біг підтюпцем	Звичайний біг

Підбиваючи підсумки, перші 5 кроків не було розпізнано через недостатню кількість даних. На цьому етапі фільтри не мали змоги класифікувати таку малу кількість рухів. Починаючи з 30 кроків, класифікація стає можливою, однак точність передбачень залишається низькою.

Найвищий результат спостерігається для випадків із 100 кроками. У цьому випадку сигнал має достатню кількість даних та чіткість, щоб забезпечити точніше передбачення. Однак залишаються труднощі з розпізнаванням

звичайного бігу та бігу підтюпцем через схожість амплітуд рухів. Саме ці види активності демонструють найнижчий рівень точності розпізнавання.

ВИСНОВКИ

У рамках магістерської дипломної роботи було розроблено нативний Android-додаток JetStep, призначений для моніторингу фізичної активності користувача.

Перед початком розробки було проведено аналіз вимог до застосунку, вивчено існуючі аналоги, а також специфіку рухів. Окрім цього, виконано порівняння наявних бібліотек, середовищ розробки та мов програмування.

Розробка Android-додатка здійснювалася в середовищі Android Studio, тоді як класифікатор було розроблено в Jupyter Notebook. У процесі роботи дотримувалися принципів Чистої Архітектури, реалізовано архітектурний патерн MVI. Завдяки цьому вдалося створити стабільну, менш схильну до багів і легше підтримувану систему.

Для класифікації та фільтрації даних використовувалися стандартні методи, такі як фільтр нижніх частот і алгоритм К-найближчих сусідів, а також було розроблено декілька власних алгоритмів, адаптованих до специфіки завдання.

Розроблена система відповідає всім вимогам, визначеним у рамках теми. Репозиторій проєкту доступний на GitHub [19].

Можливі напрями покращення додатка:

- додати функціонал для перегляду статистики кроків за днями;
- розширити можливості додатка підтримкою нового виду активності – підйом сходами;
- зібрати більше даних для покращення навчання моделі;
- додати підтримку класифікації кроків із телефоном у руках.

ПЕРЕЛІК ДЖЕРЕЛ І ПОСИЛАННЯ

1. Google Fit [Electronic resource]. – Access mode: <https://support.google.com/fit/answer/10066680?hl=en&co=GENIE.Platform%3DAndroid>
2. Samsung Health [Electronic resource]. – Access mode: https://uk.wikipedia.org/wiki/Samsung_Health
3. Android Motion Sensors [Electronic resource]. – Access mode: https://developer.android.com/develop/sensors-and-location/sensors/sensors_motion
4. Park J. Synthesis of natural arm swing motion in human bipedal walking [Text] / J. Park // Journal of Biomechanics. – 2008. – C. 2-6 – PMID: 18417138.
5. Motion Mode Recognition and Step Detection Algorithms for Mobile Phone Users [Text] / [M. Susi, V. Renaudin, G. Lachapelle] // Multidisciplinary Digital Publishing Institute (MDPI) – 2013. – C. 1540-1559 – ISSN: 1424-8220.
6. Euclidean space [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Euclidean_space#Euclidean_norm
7. Kotlin vs. Java for Android development. Levi Vaguez. August 17, 2022 LogRocket Blog [Electronic resource]. – Access mode: <https://blog.logrocket.com/kotlin-vs-java-android-development/>
8. Groovy vs Kotlin DSL. Sertaç Ayhan. Dec 15, 2023 [Electronic resource]. – Access mode: <https://blog.logrocket.com/kotlin-vs-java-android-development/>
9. What to choose Realm or SQLite with Room? Mukesh Solanki. Mar 14, 2022 [Electronic resource]. – Access mode: <https://itnext.io/what-to-choose-realm-or-sqlite-with-room-e55c34b1675c>
10. Hilt vs Koin. Yigitkaantonkaz. [Electronic resource]. – Access mode: <https://medium.com/ibtech/hilt-vs-koin-f3532b5796eb>
11. Google Fit APIs end of service overview. [Electronic resource]. – Access mode: <https://developer.android.com/health-and-fitness/guides/health-connect/migrate/fit-apis-end-of-service>

12. Fitbit OAuth 2.0 Tutorial. [Electronic resource]. – Access mode: <https://dev.fitbit.com/build/reference/web-api/troubleshooting-guide/oauth2-tutorial/>
13. Run or Walk (Part 2): Collecting Device Motion Data the Right Way. Viktor Malyi. Aug 14, 2017 [Electronic resource]. – Access mode: <https://towardsdatascience.com/run-or-walk-part-2-collecting-device-motion-data-the-right-way-58a277ff2087>
14. The Clean Architecture. Robert C. Martin. 13 August 2012. [Electronic resource]. – Access mode: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>
15. MVI Architecture with Android. Rim Gazzah. Nov 2, 2020. [Electronic resource]. – Access mode: <https://medium.com/swlh/mvi-architecture-with-android-fcde123e3c4a>
16. Why adopt Compose [Electronic resource]. – Access mode: <https://developer.android.com/develop/ui/compose/why-adopt>
17. Material Design 3 [Electronic resource]. – Access mode: <https://m3.material.io/get-started>
18. Guide to Foreground Services on Android 14. [Electronic resource]. – Access mode: <https://medium.com/@domen.lanisnik/guide-to-foreground-services-on-android-9d0127dc8f9a>
19. Github JetStep [Electronic resource]. – Access mode: <https://github.com/ZhEgor/JetStep>

ДОДАТОК А**Код програми**

```

# Gradle properties are not passed to included builds
https://github.com/gradle/gradle/issues/2534
org.gradle.parallel=true
org.gradle.caching=true
org.gradle.configureondemand=true

dependencyResolutionManagement {
    repositories {
        google()
        mavenCentral()
    }
    versionCatalogs {
        create("libs") {
            from(files("../gradle/libs.versions.toml"))
        }
    }
}

rootProject.name = "build-logic"
include(":convention")

import org.jetbrains.kotlin.gradle.dsl.JvmTarget

plugins {
    `kotlin-dsl`
}

group = "com.zhiro.jetstep.build.logic.convention"

// Configure the build-logic plugins to target JDK 17
// This matches the JDK used to build the project, and is not related to what is
// running on device.
java {
    sourceCompatibility = JavaVersion.VERSION_17
    targetCompatibility = JavaVersion.VERSION_17
}

kotlin {
    compilerOptions {
        jvmTarget = JvmTarget.JVM_17
    }
}

```

```

dependencies {
    compileOnly(libs.android.gradlePlugin)
    compileOnly(libs.kotlin.gradlePlugin)
    compileOnly(libs.androidx.room.gradlePlugin)
    compileOnly(libs.kotlin.ksp.gradlePlugin)
    compileOnly(libs.compose.gradlePlugin)
}

gradlePlugin {
    plugins {
        register("androidApplicationCompose") {
            id = "jetstep.android.application.compose"
            implementationClass = "AndroidApplicationComposeConventionPlugin"
        }
        register("androidApplication") {
            id = "jetstep.android.application"
            implementationClass = "AndroidApplicationConventionPlugin"
        }
        register("androidLibraryCompose") {
            id = "jetstep.android.library.compose"
            implementationClass = "AndroidLibraryComposeConventionPlugin"
        }
        register("androidLibrary") {
            id = "jetstep.android.library"
            implementationClass = "AndroidLibraryConventionPlugin"
        }
        register("androidFeature") {
            id = "jetstep.android.feature"
            implementationClass = "AndroidFeatureConventionPlugin"
        }
        register("jvmLibrary") {
            id = "jetstep.jvm.library"
            implementationClass = "JvmLibraryConventionPlugin"
        }
        register("daggerLibrary") {
            id = "jetstep.dagger.library"
            implementationClass = "DaggerLibraryConventionPlugin"
        }
        register("koin") {
            id = "jetstep.koin"
            implementationClass = "KoinLibraryConventionPlugin"
        }
    }
}

```

```

        register("androidKoin") {
            id = "jetstep.android.koin"
            implementationClass = "AndroidKoinLibraryConventionPlugin"
        }
        register("flavors") {
            id = "jetstep.flavors"
            implementationClass = "AndroidApplicationFlavorsConventionPlugin"
        }
        register("roomLibrary") {
            id = "jetstep.room.library"
            implementationClass = "RoomLibraryConventionPlugin"
        }
        register("detekt") {
            id = "jetstep.detekt"
            implementationClass = "DetektConventionPlugin"
        }
    }
}

import com.android.build.api.dsl.ApplicationExtension
import com.zhiro.jetstep.build.logic.convention.configureAndroidCompose
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.getByType

class AndroidApplicationComposeConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            pluginManager.apply("com.android.application")
            pluginManager.apply("org.jetbrains.kotlin.plugin.compose")
            val extension = extensions.getByType<ApplicationExtension>()
            configureAndroidCompose(extension)
        }
    }
}

import com.android.build.api.dsl.ApplicationExtension
import com.android.build.gradle.internal.dsl.BaseAppModuleExtension
import com.zhiro.jetstep.build.logic.convention.configureApkName
import com.zhiro.jetstep.build.logic.convention.configureKotlinAndroid
import com.zhiro.jetstep.build.logic.convention.configureSigning
import com.zhiro.jetstep.build.logic.convention.utils.Libs
import org.gradle.api.Plugin

```

```

import org.gradle.api.Project
import org.gradle.kotlin.dsl.configure

class AndroidApplicationConventionPlugin : Plugin<Project> {

    override fun apply(target: Project) {
        with(target) {
            with(pluginManager) {
                apply("com.android.application")
                apply("org.jetbrains.kotlin.android")
                apply("jetstep.flavors")
                apply("jetstep.detekt")
            }

            extensions.configure<ApplicationExtension> {
                configureKotlinAndroid(this)
                configureSigning(this)
                defaultConfig.targetSdk =
libs.findVersion("targetSdk").get().toString().toInt()
            }

            extensions.configure<BaseAppModuleExtension> {
                configureApkName(this)
            }
        }
    }
}

import com.android.build.api.dsl.ApplicationExtension
import com.zhiro.jetstep.build.logic.convention.configureFlavors
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.configure

class AndroidApplicationFlavorsConventionPlugin : Plugin<Project> {

    override fun apply(target: Project) {
        with(target) {
            extensions.configure<ApplicationExtension> {
                configureFlavors(this)
            }
        }
    }
}

```

```

    }
}

import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.dependencies
import org.gradle.kotlin.dsl.kotlin

class AndroidFeatureConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            pluginManager.apply {
                apply("jetstep.android.library")
                apply("jetstep.android.koin")
                apply("jetstep.android.library.compose")
            }

            dependencies {
                add("implementation", project(":domain"))
                add("implementation", project(":core:ui"))
                add("implementation", project(":core:navigation"))
                add("implementation", project(":core:common"))

                add("testImplementation", project(":domain-test"))
                add("androidTestImplementation", project(":domain-test"))

                add("testImplementation", kotlin("test"))
                add("androidTestImplementation", kotlin("test"))
            }
        }
    }
}

import com.zhiro.jetstep.build.logic.convention.configureKoinAndroid
import org.gradle.api.Plugin
import org.gradle.api.Project

class AndroidKoinLibraryConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            configureKoinAndroid()
        }
    }
}

```

```

    }
}

import com.android.build.gradle.LibraryExtension
import com.zhiro.jetstep.build.logic.convention.configureAndroidCompose
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.getByType

class AndroidLibraryComposeConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            pluginManager.apply("com.android.library")
            pluginManager.apply("org.jetbrains.kotlin.plugin.compose")
            val extension = extensions.getByType<LibraryExtension>()
            configureAndroidCompose(extension)
        }
    }
}

import com.android.build.api.dsl.LibraryExtension
import com.zhiro.jetstep.build.logic.convention.configureFlavors
import com.zhiro.jetstep.build.logic.convention.configureKotlinAndroid
import com.zhiro.jetstep.build.logic.convention.utils.libs
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.configure
import org.gradle.kotlin.dsl.dependencies
import org.gradle.kotlin.dsl.kotlin

class AndroidLibraryConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            with(pluginManager) {
                apply("com.android.library")
                apply("org.jetbrains.kotlin.android")
                apply("org.jetbrains.kotlin.plugin.serialization")
                apply("jetstep.detekt")
            }

            extensions.configure<LibraryExtension> {
                configureKotlinAndroid(this)
                configureFlavors(this)
            }
        }
    }
}

```

```

    }

    dependencies {
        add("implementation", libs.findLibrary("kotlin-serialization-
json").get())
        add("androidTestImplementation", kotlin("test"))
        add("testImplementation", kotlin("test"))
    }
}

}

import com.zhiro.jetstep.build.logic.convention.utils.libs
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.dependencies

class DaggerLibraryConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            pluginManager.apply {
                apply("com.google.devtools.ksp")
            }

            dependencies {
                add("implementation", libs.findLibrary("dagger").get())
            }
        }
    }
}

import org.gradle.api.Plugin
import org.gradle.api.Project

class DetektConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            pluginManager.apply {
                apply("io.gitlab.arturbosch.detekt")
            }
        }
    }
}

```

```
import com.zhiro.jetstep.build.logic.convention.configureKotlinJvm
import org.gradle.api.Plugin
import org.gradle.api.Project
```

```
class JvmLibraryConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            with(pluginManager) {
                apply("org.jetbrains.kotlin.jvm")
            }
            configureKotlinJvm()
        }
    }
}
```

```
import com.zhiro.jetstep.build.logic.convention.configureKoin
import org.gradle.api.Plugin
import org.gradle.api.Project
```

```
class KoinLibraryConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            configureKoin()
        }
    }
}
```

```
import androidx.room.gradle.RoomExtension
import com.google.devtools.ksp.gradle.KspExtension
import com.zhiro.jetstep.build.logic.convention.utils.libs
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.configure
import org.gradle.kotlin.dsl.dependencies
```

```
class RoomLibraryConventionPlugin : Plugin<Project> {
    override fun apply(target: Project) {
        with(target) {
            with(pluginManager) {
                apply("com.google.devtools.ksp")
                apply("androidx.room")
            }
        }
    }
}
```

```

        dependencies {
            add("implementation", libs.findLibrary("androidx-room-
runtime").get())
            add("implementation", libs.findLibrary("androidx-room-
ktx").get())
            add("ksp", libs.findLibrary("androidx-room-compiler").get())
        }
        extensions.configure<KspExtension> {
            arg("room.generateKotlin", "true")
        }
        extensions.configure<RoomExtension> {
            schemaDirectory("$projectDir/schemas")
        }
    }
}
}

```

#Tue Oct 08 17:54:57 EEST 2024

distributionBase=GRADLE_USER_HOME

distributionPath=wrapper/dists

distributionUrl=https\://services.gradle.org/distributions/gradle-8.9-bin.zip

zipStoreBase=GRADLE_USER_HOME

zipStorePath=wrapper/dists

[versions]

agp = "8.7.0"

app = "8.1.4"

kotlin = "2.0.21"

ksp = "2.0.21-1.0.25"

coroutines = "1.9.0"

coreKtx = "1.13.1"

junit = "4.13.2"

junitVersion = "1.2.1"

espressoCore = "3.6.1"

lifecycleRuntimeKtx = "2.8.6"

activityCompose = "1.9.2"

composeBom = "2024.09.03"

appcompat = "1.7.0"

material = "1.12.0"

koin = "4.0.0"

navigation = "2.8.2"

room = "2.6.1"

detekt = "1.21.0"

```

serializationJson = "1.7.3"
datastore = "1.1.1"
onnxruntime = "1.20.0"

```

```

# android sdk start
minSdk = "27"
compileSdk = "35"
targetSdk = "35"
# android sdk end

```

```

[libraries]
androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref =
"coreKtx" }
junit = { group = "junit", name = "junit", version.ref = "junit" }
androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref =
"junitVersion" }
androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso-
core", version.ref = "espressoCore" }
androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name =
"lifecycle-runtime-ktx", version.ref = "lifecycleRuntimeKtx" }
androidx-activity-compose = { group = "androidx.activity", name = "activity-
compose", version.ref = "activityCompose" }
androidx-compose-bom = { group = "androidx.compose", name = "compose-bom",
version.ref = "composeBom" }
androidx-ui = { group = "androidx.compose.ui", name = "ui" }
androidx-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" }
androidx-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" }
androidx-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui-
tooling-preview" }
androidx-ui-test-manifest = { group = "androidx.compose.ui", name = "ui-test-
manifest" }
androidx-ui-test-junit4 = { group = "androidx.compose.ui", name = "ui-test-
junit4" }
androidx-material3 = { group = "androidx.compose.material3", name = "material3"
}
androidx-appcompat = { group = "androidx.appcompat", name = "appcompat",
version.ref = "appcompat" }
material = { group = "com.google.android.material", name = "material",
version.ref = "material" }
kotlin-coroutines-core = { group = "org.jetbrains.kotlinx", name = "kotlinx-

```

```

coroutines-core", version.ref = "coroutines" }
kotlin-serialization-json = { group = "org.jetbrains.kotlinx", name = "kotlinx-
serialization-json", version.ref = "serializationJson" }
androidx-navigation-compose = { group = "androidx.navigation", name =
"navigation-compose", version.ref = "navigation" }
androidx-room-runtime = { group = "androidx.room", name = "room-runtime",
version.ref = "room" }
androidx-room-compiler = { group = "androidx.room", name = "room-compiler",
version.ref = "room" }
androidx-room-ktx = { group = "androidx.room", name = "room-ktx", version.ref =
"room" }
androidx-room-testing = { group = "androidx.room", name = "room-testing",
version.ref = "room" }
androidx-datastore-preferences = { group = "androidx.datastore", name =
"datastore-preferences", version.ref = "datastore"}
onnxruntime-android = { group = "com.microsoft.onnxruntime", name =
"onnxruntime-android", version.ref = "onnxruntime"}

android-gradlePlugin = { group = "com.android.tools.build", name = "gradle",
version.ref = "app" }
kotlin-gradlePlugin = { group = "org.jetbrains.kotlin", name = "kotlin-gradle-
plugin", version.ref = "kotlin" }
androidx-room-gradlePlugin = { group = "androidx.room", name = "room-gradle-
plugin", version.ref = "room" }
kotlin-ksp-gradlePlugin = { group = "com.google.devtools.ksp", name =
"com.google.devtools.ksp.gradle.plugin", version.ref = "ksp" }
compose-gradlePlugin = { group = "org.jetbrains.kotlin", name = "compose-
compiler-gradle-plugin", version.ref = "kotlin" }
koin-bom = { group = "io.insert-koin", name = "koin-bom", version.ref = "koin" }
koin-core = { group = "io.insert-koin", name = "koin-core" }
koin-android = { group = "io.insert-koin", name = "koin-android" }
koin-androidx-compose = { group = "io.insert-koin", name = "koin-androidx-
compose" }

```

[plugins]

```

android-application = { id = "com.android.application", version.ref = "app" }
kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
kotlin-compose = { id = "org.jetbrains.kotlin.plugin.compose", version.ref =
"kotlin" }
android-library = { id = "com.android.library", version.ref = "app" }

```

```

jetbrains-kotlin-jvm = { id = "org.jetbrains.kotlin.jvm", version.ref = "kotlin"
}
kotlin-ksp = { id = "com.google.devtools.ksp", version.ref = "ksp" }
kotlin-serialization = { id = "org.jetbrains.kotlin.plugin.serialization",
version.ref = "kotlin" }
room = { id = "androidx.room", version.ref = "room" }
detekt = { id = "io.gitlab.arturbosch.detekt", version.ref = "detekt" }

# Plugins defined by this project
jetstep-android-application = { id = "jetstep.android.application" }
jetstep-android-application-compose = { id =
"jetstep.android.application.compose" }
jetstep-android-feature = { id = "jetstep.android.feature" }
jetstep-android-library = { id = "jetstep.android.library" }
jetstep-android-library-compose = { id = "jetstep.android.library.compose" }
jetstep-jvm-library = { id = "jetstep.jvm.library" }
jetstep-room-library = { id = "jetstep.room.library" }
jetstep-detekt = { id = "jetstep.detekt" }
jetstep-android-koin = { id = "jetstep.android.koin" }
jetstep-koin = { id = "jetstep.koin" }

// Top-level build file where you can add configuration options common to all
sub-projects/modules.
plugins {
    alias(libs.plugins.android.application) apply false
    alias(libs.plugins.android.library) apply false
    alias(libs.plugins.kotlin.android) apply false
    alias(libs.plugins.kotlin.compose) apply false
    alias(libs.plugins.jetbrains.kotlin.jvm) apply false
    alias(libs.plugins.kotlin.ksp) apply false
    alias(libs.plugins.kotlin.serialization) apply false
    alias(libs.plugins.room) apply false
    alias(libs.plugins.detekt) apply false
}

// This file contains classes (with possible wildcards) that the Compose
Compiler will treat as stable.
// It allows us to define classes that are not part of our codebase without
wrapping them in a stable class.
// For more information, check
https://developer.android.com/jetpack/compose/performance/stability/fix#configuration-file

```

```
// We always use immutable classes for our data model, to avoid running the
Compose compiler
// in the module we declare it to be stable here.
// com.zhiro.jetstep.domain-model.*

// Java standard library classes
java.time.ZoneId
java.time.ZoneOffset

// Kotlin collections classes
kotlin.collections.List
kotlin.collections.Set
kotlin.collections.Map

# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build\_environment.html
# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. For more details,
visit
# https://developer.android.com/r/tools/gradle-multi-project-decoupled-projects
# org.gradle.parallel=true
# AndroidX package structure to make it clearer which packages are bundled with
the
# Android operating system, and which are packaged with your app's APK
# https://developer.android.com/topic/libraries/support-library/androidx-rn
android.useAndroidX=true
# Kotlin code style for this project: "official" or "obsolete":
kotlin.code.style=official
# Enables namespacing of each library's R class so that its R class includes
only the
# resources declared in the library itself and none from the library's
dependencies,
# thereby reducing the size of the R class for that library
android.nonTransitiveRClass=true
```

```
#!/usr/bin/env sh
```

```
#
# Copyright 2015 the original author or authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
```

```
#####
```

```
##
```

```
## Gradle start up script for UN*X
```

```
##
```

```
#####
```

```
# Attempt to set APP_HOME
# Resolve links: $0 may be a link
PRG="$0"
# Need this for relative symlinks.
while [ -h "$PRG" ] ; do
    ls=`ls -ld "$PRG"`
    link=`expr "$ls" : '.*-> \(.*)$'`
    if expr "$link" : '/.*' > /dev/null; then
        PRG="$link"
    else
        PRG=`dirname "$PRG"`"/$link"
    fi
done
SAVED="`pwd`"
cd "`dirname \"$PRG\"`/" >/dev/null
APP_HOME="`pwd -P`"
cd "$SAVED" >/dev/null
```

```

APP_NAME="Gradle"
APP_BASE_NAME=`basename "$0"`

# Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to
# pass JVM options to this script.
DEFAULT_JVM_OPTS='"-Xmx64m" "-Xms64m"'

# Use the maximum available, or set MAX_FD != -1 to use that value.
MAX_FD="maximum"

warn () {
    echo "$*"
}

die () {
    echo
    echo "$*"
    echo
    exit 1
}

# OS specific support (must be 'true' or 'false').
cygwin=false
msys=false
darwin=false
nonstop=false
case "`uname`" in
    CYGWIN* )
        cygwin=true
        ;;
    Darwin* )
        darwin=true
        ;;
    MINGW* )
        msys=true
        ;;
    NONSTOP* )
        nonstop=true
        ;;
esac

CLASSPATH=$APP_HOME/gradle/wrapper/gradle-wrapper.jar

```

```

# Determine the Java command to use to start the JVM.
if [ -n "$JAVA_HOME" ] ; then
    if [ -x "$JAVA_HOME/jre/sh/java" ] ; then
        # IBM's JDK on AIX uses strange locations for the executables
        JAVACMD="$JAVA_HOME/jre/sh/java"
    else
        JAVACMD="$JAVA_HOME/bin/java"
    fi
    if [ ! -x "$JAVACMD" ] ; then
        die "ERROR: JAVA_HOME is set to an invalid directory: $JAVA_HOME

```

Please set the JAVA_HOME variable in your environment to match the location of your Java installation."

```

    fi
else
    JAVACMD="java"
    which java >/dev/null 2>&1 || die "ERROR: JAVA_HOME is not set and no 'java'
command could be found in your PATH.

```

Please set the JAVA_HOME variable in your environment to match the location of your Java installation."

```

fi

# Increase the maximum file descriptors if we can.
if [ "$cygwin" = "false" -a "$darwin" = "false" -a "$nonstop" = "false" ] ; then
    MAX_FD_LIMIT=`ulimit -H -n`
    if [ $? -eq 0 ] ; then
        if [ "$MAX_FD" = "maximum" -o "$MAX_FD" = "max" ] ; then
            MAX_FD="$MAX_FD_LIMIT"
        fi
        ulimit -n $MAX_FD
        if [ $? -ne 0 ] ; then
            warn "Could not set maximum file descriptor limit: $MAX_FD"
        fi
    else
        warn "Could not query maximum file descriptor limit: $MAX_FD_LIMIT"
    fi
fi

```

For Darwin, add options to specify how the application appears in the dock

```

if $darwin; then
    GRADLE_OPTS="$GRADLE_OPTS \"-Xdock:name=$APP_NAME\" \"-
Xdock:icon=$APP_HOME/media/gradle.icns\""
fi

# For Cygwin or MSYS, switch paths to Windows format before running java
if [ "$cygwin" = "true" -o "$msys" = "true" ] ; then
    APP_HOME=`cygpath --path --mixed "$APP_HOME"`
    CLASSPATH=`cygpath --path --mixed "$CLASSPATH"`

    JAVACMD=`cygpath --unix "$JAVACMD"`

    # We build the pattern for arguments to be converted via cygpath
    ROOTDIRSRAW=`find -L / -maxdepth 1 -mindepth 1 -type d 2>/dev/null`
    SEP=""
    for dir in $ROOTDIRSRAW ; do
        ROOTDIRS="$ROOTDIRS$SEP$dir"
        SEP="|"
    done
    OURCYGPATTERN="(^($ROOTDIRS))"
    # Add a user-defined pattern to the cygpath arguments
    if [ "$GRADLE_CYGPATTERN" != "" ] ; then
        OURCYGPATTERN="$OURCYGPATTERN|($GRADLE_CYGPATTERN)"
    fi
    # Now convert the arguments - kludge to limit ourselves to /bin/sh
    i=0
    for arg in "$@" ; do
        CHECK=`echo "$arg"|egrep -c "$OURCYGPATTERN" -`
        CHECK2=`echo "$arg"|egrep -c "^-"`           ###
        Determine if an option

        if [ $CHECK -ne 0 ] && [ $CHECK2 -eq 0 ] ; then        ###
            Added a condition
            eval `echo args$i`=`cygpath --path --ignore --mixed "$arg"`
        else
            eval `echo args$i`="\"$arg\""
        fi
        i=`expr $i + 1`
    done
    case $i in
        0) set -- ;;
        1) set -- "$args0" ;;

```

```

2) set -- "$args0" "$args1" ;;
3) set -- "$args0" "$args1" "$args2" ;;
4) set -- "$args0" "$args1" "$args2" "$args3" ;;
5) set -- "$args0" "$args1" "$args2" "$args3" "$args4" ;;
6) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" ;;
7) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6"
;;
8) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6"
"$args7" ;;
9) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6"
"$args7" "$args8" ;;
esac
fi

# Escape application args
save () {
    for i do printf %s\\n "$i" | sed "s/'/'\\\\\\'/g;1s/^/'/;\\$s/\\$/' \\\\\\\'/ " ;
done
    echo " "
}
APP_ARGS=`save "$@"`

# Collect all arguments for the java command, following the shell quoting and
substitution rules
eval set -- $DEFAULT_JVM_OPTS $JAVA_OPTS $GRADLE_OPTS "\"-
Dorg.gradle.appname=$APP_BASE_NAME\""" -classpath "\"$CLASSPATH\"""
org.gradle.wrapper.GradleWrapperMain "$APP_ARGS"

exec "$JAVACMD" "$@"

@rem
@rem Copyright 2015 the original author or authors.
@rem
@rem Licensed under the Apache License, Version 2.0 (the "License");
@rem you may not use this file except in compliance with the License.
@rem You may obtain a copy of the License at
@rem
@rem      https://www.apache.org/licenses/LICENSE-2.0
@rem
@rem Unless required by applicable law or agreed to in writing, software
@rem distributed under the License is distributed on an "AS IS" BASIS,
@rem WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
@rem See the License for the specific language governing permissions and

```

```

@rem limitations under the License.
@rem

@if "%DEBUG%" == "" @echo off
@rem #####
@rem
@rem Gradle startup script for Windows
@rem
@rem #####

@rem Set local scope for the variables with windows NT shell
if "%OS%"=="Windows_NT" setlocal

set DIRNAME=%~dp0
if "%DIRNAME%" == "" set DIRNAME=.
set APP_BASE_NAME=%~n0
set APP_HOME=%DIRNAME%

@rem Resolve any "." and ".." in APP_HOME to make it shorter.
for %%i in ("%APP_HOME%") do set APP_HOME=%%~fi

@rem Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to
pass JVM options to this script.
set DEFAULT_JVM_OPTS="-Xmx64m" "-Xms64m"

@rem Find java.exe
if defined JAVA_HOME goto findJavaFromJavaHome

set JAVA_EXE=java.exe
%JAVA_EXE% -version >NUL 2>&1
if "%ERRORLEVEL%" == "0" goto execute

echo.
echo ERROR: JAVA_HOME is not set and no 'java' command could be found in your
PATH.
echo.
echo Please set the JAVA_HOME variable in your environment to match the
echo location of your Java installation.

goto fail

:findJavaFromJavaHome

```

```

set JAVA_HOME=%JAVA_HOME:="%
set JAVA_EXE=%JAVA_HOME%/bin/java.exe

if exist "%JAVA_EXE%" goto execute

echo.
echo ERROR: JAVA_HOME is set to an invalid directory: %JAVA_HOME%
echo.
echo Please set the JAVA_HOME variable in your environment to match the
echo location of your Java installation.

goto fail

:execute
@rem Setup the command line

set CLASSPATH=%APP_HOME%\gradle\wrapper\gradle-wrapper.jar


@rem Execute Gradle
"%JAVA_EXE%" %DEFAULT_JVM_OPTS% %JAVA_OPTS% %GRADLE_OPTS% "-
Dorg.gradle.appname=%APP_BASE_NAME%" -classpath "%CLASSPATH%"
org.gradle.wrapper.GradleWrapperMain %*

:end
@rem End local scope for the variables with windows NT shell
if "%ERRORLEVEL%"=="0" goto mainEnd

:fail
rem Set variable GRADLE_EXIT_CONSOLE if you need the _script_ return code
instead of
rem the _cmd.exe /c_ return code!
if not "" == "%GRADLE_EXIT_CONSOLE%" exit 1
exit /b 1

:mainEnd
if "%OS%"=="Windows_NT" endlocal

:omega

pluginManagement {
    includeBuild("build-logic")
    repositories {

```

```

        google {
            content {
                includeGroupByRegex("com\\.\\.android.*")
                includeGroupByRegex("com\\.\\.google.*")
                includeGroupByRegex("androidx.*")
            }
        }
        mavenCentral()
        gradlePluginPortal()
    }
}

dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
    }
}

rootProject.name = "JetStep"
enableFeaturePreview("TYPESAFE_PROJECT_ACCESSORS")
include(":app")
include(":feature:dashboard")
include(":core:common")
include(":core:design-system")
include(":core:navigation")
include(":core:ui")
include(":domain")
include(":domain-test")
include(":domain-model")
include(":data")
include(":feature:locomotionclassifier")

plugins {
    alias(libs.plugins.jetstep.android.application)
    alias(libs.plugins.jetstep.android.application.compose)
    alias(libs.plugins.jetstep.android.koin)
}

android {
    namespace = "com.zhiro.jetstep"

    defaultConfig {

```

```

        applicationId = "com.zhiro.jetstep"
        versionCode = 1
        versionName = "0.0.1" // X.Y.Z; X = Major, Y = Minor, Z = Patch level

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        debug {
            signingConfig = signingConfigs.named("debug").get()
        }
        release {
            isDebuggable = false
            isMinifyEnabled = true
            isShrinkResources = true
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
            signingConfig = signingConfigs.named("release").get()
        }
    }
}

dependencies {
    implementation/projects/domain
    implementation/projects/core/designSystem
    implementation/projects/core/navigation
    implementation/projects/core/common
    implementation/projects/feature/dashboard
    implementation/projects/feature/locomotionclassifier

    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(libs.androidx.ui.test.junit4)
}

package com.zhiro.jetstep.android.activities

import android.annotation.SuppressLint
import android.app.ActivityManager
import android.content.Intent

```

```

import android.content.pm.PackageManager
import android.os.Build
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.runtime.Composable
import androidx.core.content.ContextCompat
import androidx.navigation.compose.rememberNavController
import com.zhiro.jetstep.android.services.FitnessForegroundService
import com.zhiro.jetstep.core.design.system.theme.JetStepTheme
import com.zhiro.jetstep.core.navigation.utils.InitNavigationManager
import com.zhiro.jetstep.navigation.NavigationGraph

class MainActivity : ComponentActivity() {

    @SuppressLint("InlinedApi")
    private val permissions = setOfNotNull(
        android.Manifest.permission.POST_NOTIFICATIONS.takeIf {
            Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU },
        android.Manifest.permission.ACTIVITY_RECOGNITION
    )
    private val notificationPermissionLauncher = registerForActivityResult(
        ActivityResultContracts.RequestPermission()
    ) { isGranted ->
        if (isGranted) {
            checkAndRequestNotificationPermission(
                permissions = permissions,
                onGrantedPermission = ::startBackgroundWork
            )
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent { JetStepApp() }
        checkAndRequestNotificationPermission(
            permissions = permissions,
            onGrantedPermission = ::startBackgroundWork
        )
    }
}

```

```

    }

    private fun checkAndRequestNotificationPermission(
        permissions: Set<String>,
        onGrantedPermission: () -> Unit
    ) {
        val isGranted = !permissions.any { permission ->
            if (ContextCompat.checkSelfPermission(this, permission) !=
PackageManager.PERMISSION_GRANTED) {
                notificationPermissionLauncher.launch(permission)
                true
            } else {
                false
            }
        }
        if (isGranted) {
            onGrantedPermission.invoke()
        }
    }

    private fun startBackgroundWork() {
        if (!isServiceRunning(FitnessForegroundService::class.java.name)) {
            startForegroundService(Intent(this,
FitnessForegroundService::class.java))
        }
    }

    private fun isServiceRunning(serviceName: String): Boolean {
        val activityManager = getSystemService(ACTIVITY_SERVICE) as
ActivityManager
        val services = activityManager.getRunningServices(Int.MAX_VALUE)
        return services.any { service -> service.service.className ==
serviceName }
    }
}

@Composable
private fun JetStepApp() {
    val navController = rememberNavController()

    JetStepTheme {
        InitNavigationManager(navController = navController)
    }
}

```

```

        NavigationGraph(navController = navController)
    }
}

package com.zhiro.jetstep.android.activities

import android.content.ComponentName
import android.content.Context
import android.content.Intent
import android.content.ServiceConnection
import android.os.Bundle
import android.os.IBinder
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.Button
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.rememberCoroutineScope
import androidx.compose.runtime.setValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.zhiro.jetstep.android.services.AccelerometerCollectorService
import com.zhiro.jetstep.android.services.latestValue
import com.zhiro.jetstep.core.design.system.theme.JetStepTheme
import com.zhiro.jetstep.domain.model.RawAccelerometerRecord
import com.zhiro.jetstep.domain.repository.SensorRepository
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.flow.map
import kotlinx.coroutines.launch

```

```

import org.koin.android.ext.android.inject

class SupportiveActivity : ComponentActivity() {

    private val sensorRepository: SensorRepository by inject()
    private val accelerometerRecordsFlow =
sensorRepository.getAllRawAccelerometerRecordsFlow()
        .map(List<RawAccelerometerRecord>::reversed)
    private var accelerometerCollectorService: AccelerometerCollectorService? =
null
    var bound by mutableStateOf(false)
    private val connection = object : ServiceConnection {
        override fun onServiceConnected(className: ComponentName, service:
IBinder) {
            // We've bound to LocalService, cast the IBinder and get
LocalService instance.
            val binder = service as
AccelerometerCollectorService.AccelerometerCollectorBinder
            accelerometerCollectorService = binder.getService()
            bound = true
        }

        override fun onServiceDisconnected(arg0: ComponentName) {
            bound = false
            accelerometerCollectorService?.stopSelf()
            accelerometerCollectorService = null
        }

        override fun onBindingDied(name: ComponentName?) {
            super.onBindingDied(name)
            bound = false
        }

        override fun onNullBinding(name: ComponentName?) {
            super.onNullBinding(name)
            bound = false
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
    }
}

```

```

setContent {
    val accelerometerRecords by accelerometerRecordsFlow.collectAsState(
        emptyList(),
        Dispatchers.IO
    )
    val scope = rememberCoroutineScope()

    JetStepTheme {
        Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
            Column(
                modifier = Modifier
                    .padding(innerPadding)
                    .padding(20.dp),
                verticalArrangement = Arrangement.spacedBy(8.dp)
            ) {
                Text(text = "latest value: ${latestValue.value - 9.8}",
                    fontSize = 28.sp)

                Button(
                    modifier = Modifier.fillMaxWidth(),
                    onClick = {
                        bindService(
                            Intent(
                                this@SupportiveActivity,
                                AccelerometerCollectorService::class.java
                                    ).apply {
                    putExtra(AccelerometerCollectorService.PACE_ARG, "walking")
                },
                    connection,
                    Context.BIND_AUTO_CREATE
                )
            },
                    enabled = !bound
                ) {
                    Text(text = "Start collecting walking")
                }
                Button(
                    modifier = Modifier.fillMaxWidth(),
                    onClick = {
                        bindService(

```

```

        Intent(
            this@SupportiveActivity,

AccelerometerCollectorService::class.java
        ).apply {

putExtra(AccelerometerCollectorService.PACE_ARG, "running")
        },
        connection,
        Context.BIND_AUTO_CREATE
    )
    },
    enabled = !bound
) {
    Text(text = "Start collecting running")
}
Button(
    modifier = Modifier.fillMaxWidth(),
    onClick = {
        bindService(
            Intent(
                this@SupportiveActivity,

AccelerometerCollectorService::class.java
            ).apply {

putExtra(AccelerometerCollectorService.PACE_ARG, "jogging")
        },
        connection,
        Context.BIND_AUTO_CREATE
    )
    },
    enabled = !bound
) {
    Text(text = "Start collecting jogging")
}

Button(
    modifier = Modifier.fillMaxWidth(),
    onClick = {
        scope.launch(Dispatchers.IO) {
            accelerometerCollectorService?.finish()

```



```
package com.zhiro.jetstep.android.notifications

import android.app.Notification
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.Service
import android.content.Context
import androidx.core.app.NotificationCompat
import com.zhiro.jetstep.R

internal object NotificationsHelper {

    private const val NOTIFICATION_CHANNEL_ID = "general_notification_channel"

    fun createNotificationChannel(context: Context) {
        val notificationManager =
            context.getSystemService(Service.NOTIFICATION_SERVICE) as NotificationManager

        val channel = NotificationChannel(
            NOTIFICATION_CHANNEL_ID,
            context.getString(R.string.fitness_notification_channel_name),
            NotificationManager.IMPORTANCE_DEFAULT
        )

        notificationManager.createNotificationChannel(channel)
    }

    fun buildNotification(context: Context): Notification {
        return NotificationCompat.Builder(context, NOTIFICATION_CHANNEL_ID)

            .setContentTitle(context.getString(R.string.fitness_notification_title))
    }
}
```

```

        .setSmallIcon(R.drawable.ic_launcher_foreground)

        .setForegroundServiceBehavior(NotificationCompat.FOREGROUND_SERVICE_IMMEDIATE)

        .build()
    }
}

package com.zhiro.jetstep.android.services

import android.app.Service
import android.content.Intent
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import android.os.Binder
import android.os.IBinder
import androidx.compose.runtime.mutableStateOf
import com.zhiro.jetstep.domain.model.RawAccelerometerRecord
import com.zhiro.jetstep.domain.repository.SensorRepository
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.SupervisorJob
import kotlinx.coroutines.launch
import org.koin.android.ext.android.inject
import java.util.concurrent.CopyOnWriteArrayList
import kotlin.math.sqrt

var latestValue = mutableStateOf(0.0f)

class AccelerometerCollectorService : Service(), SensorEventListener {

    private var sensorManager: SensorManager? = null
    private var accelerometerRecords =
CopyOnWriteArrayList(mutableListOf<RawAccelerometerRecord>())
    private val scope = CoroutineScope(SupervisorJob() + Dispatchers.IO)
    private val sensorRepository: SensorRepository by inject()
    private val binder = AccelerometerCollectorBinder()
    private var pace: String = ""

    override fun onBind(intent: Intent?): IBinder {

```

```

        pace = intent?.getStringExtra(PACE_ARG) ?: ""
        sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
        sensorManager?.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)?.also {
sensor ->
            sensorManager?.registerListener(this, sensor,
SAMPLING_RATE_MICROSECONDS)
        }
        return binder
    }

suspend fun finish() {
    sensorManager?.unregisterListener(this)
    sensorManager = null
    sync()
}

override fun onSensorChanged(event: SensorEvent?) {
    if (event?.sensor?.type == Sensor.TYPE_ACCELEROMETER) {
        accelerometerRecords.add(
            RawAccelerometerRecord(
                x = event.values[0],
                y = event.values[1],
                z = event.values[2],
                recordTakenAt = System.currentTimeMillis(),
                pace = pace
            )
        )
        val last = accelerometerRecords.last()!!
        latestValue.value = sqrt(last.x * last.x + last.y * last.y + last.z
* last.z)
        syncIfNeeded()
    }
}

override fun onAccuracyChanged(p0: Sensor?, p1: Int) {}

suspend fun sync() {
    val accelerometerRecordsToSync = accelerometerRecords
    sensorRepository.saveRawAccelerometerRecords(records =
accelerometerRecordsToSync)
    accelerometerRecords.removeAll(accelerometerRecordsToSync)
}

```

```

private fun syncIfNeeded() {
    if (accelerometerRecords.size > SYNC_IS_NEEDED_LIMIT) {
        scope.launch {
            sync()
        }
    }
}

inner class AccelerometerCollectorBinder : Binder() {
    // Return this instance of LocalService so clients can call public
methods.
    fun getService(): AccelerometerCollectorService =
this@AccelerometerCollectorService
}

companion object {
    const val PACE_ARG = "pace_arg"
    private const val SYNC_IS_NEEDED_LIMIT = 500
    private const val SAMPLING_RATE_MICROSECONDS = 20_000 // 50 HZ
}
}

package com.zhiro.jetstep.android.services

import android.annotation.SuppressLint
import android.app.Service
import android.content.Intent
import android.content.pm.ServiceInfo
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import android.os.Build
import android.os.IBinder
import androidx.core.app.ServiceCompat
import com.zhiro.jetstep.android.notifications.NotificationsHelper
import com.zhiro.jetstep.domain.repository.StepRepository
import com.zhiro.jetstep.feature.locomotionclassifier.LocomotionHelper
import com.zhiro.jetstep.feature.locomotionclassifier.model.ClassifiedSteps
import com.zhiro.jetstep.utils.mappers.toStepBatch
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers

```

```

import kotlinx.coroutines.SupervisorJob
import kotlinx.coroutines.cancel
import kotlinx.coroutines.flow.map
import kotlinx.coroutines.launch
import org.koin.android.ext.android.inject

class FitnessForegroundService : Service(), SensorEventListener {

    val scope = CoroutineScope(Dispatchers.IO + SupervisorJob())
    private var sensorManager: SensorManager? = null
    private val locomotionHelper: LocomotionHelper by inject()
    private val stepRepository: StepRepository by inject()

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int
    {
        startAsForegroundService()
        registerSensors()
        observeClassifiedSteps()
        return super.onStartCommand(intent, flags, startId)
    }

    private fun observeClassifiedSteps() {
        scope.launch {

locomotionHelper.lastClassifiedStepsFlow.map(ClassifiedSteps::toStepBatch).collect { stepBatch ->
                stepRepository.saveStepBatch(stepBatch)
            }
        }
    }

    fun registerSensors() {
        sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
        sensorManager?.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)?.also {
sensor ->
            sensorManager?.registerListener(this, sensor,
SAMPLING_RATE_MICROSECONDS)
        }
        sensorManager?.getDefaultSensor(Sensor.TYPE_STEP_COUNTER)?.also { sensor
->
            sensorManager?.registerListener(this, sensor,
SAMPLING_RATE_MICROSECONDS)

```

```

    }
}

private fun unregisterSensors() {
    sensorManager?.unregisterListener(this)
    sensorManager = null
}

override fun onSensorChanged(event: SensorEvent?) {
    when (event?.sensor?.type) {
        Sensor.TYPE_ACCELEROMETER -> locomotionHelper.feedAccelerometer(
            x = event.values[0],
            y = event.values[1],
            z = event.values[2]
        )

        Sensor.TYPE_STEP_COUNTER -> locomotionHelper.feedSteps(
            aggregatedStepsCount = event.values[0].toLong()
        )
    }
}

@SuppressLint("ForegroundServiceType")
private fun startAsForegroundService() {
    NotificationsHelper.createNotificationChannel(this)

    ServiceCompat.startForeground(
        this,
        FOREGROUND_SERVICE_ID,
        NotificationsHelper.buildNotification(this),
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            ServiceInfo.FOREGROUND_SERVICE_TYPE_HEALTH
        } else {
            0
        }
    )
}

override fun onDestroy() {
    unregisterSensors()
    scope.cancel()
}

```

```

        super.onDestroy()
    }

    override fun onAccuracyChanged(event: Sensor?, p1: Int) {}

    override fun onBind(p0: Intent?): IBinder? = null

    companion object {
        private const val FOREGROUND_SERVICE_ID = 1
        private const val SAMPLING_RATE_MICROSECONDS = 20_000 // 50 HZ
    }
}

package com.zhiro.jetstep.navigation

import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import com.zhiro.jetstep.core.navigation.routes.ScreenRoute
import com.zhiro.jetstep.feature.dashboard.presentation.dashboard.dashboardGraph

@Composable
fun NavigationGraph(
    modifier: Modifier = Modifier,
    navController: NavHostController,
    startDestination: ScreenRoute = ScreenRoute.Dashboard,
) {
    NavHost(
        modifier = modifier,
        navController = navController,
        startDestination = startDestination
    ) {
        dashboardGraph()
    }
}

package com.zhiro.jetstep.utils.initmanager

interface InitManager {
    fun init()
}

```

```

package com.zhiro.jetstep.utils.initmanager

import android.app.Application
import com.zhiro.jetstep.core.common.di.module.commonModule
import com.zhiro.jetstep.core.navigation.di.modules.navigationModule
import com.zhiro.jetstep.domain.di.module.domainModule
import com.zhiro.jetstep.feature.dashboard.di.module.featureDashboardModule
import
com.zhiro.jetstep.feature.locomotionclassifier.di.module.featureLocomotionHelper
Module
import org.koin.android.ext.koin.androidContext
import org.koin.core.context.startKoin

class KoinInitManager(private val application: Application) : InitManager {

    override fun init() {
        startKoin {
            androidContext(application)
            modules(
                domainModule,
                commonModule,
                navigationModule,
                featureDashboardModule,
                featureLocomotionHelperModule
            )
        }
    }
}

package com.zhiro.jetstep.utils.mappers

import com.zhiro.jetstep.domain.model.StepBatch
import com.zhiro.jetstep.domain.model.StepType
import com.zhiro.jetstep.feature.locomotionclassifier.model.ClassifiedSteps

fun ClassifiedSteps.toStepBatch() = StepBatch(
    steps = steps.toInt(),
    startTime = startTime,
    endTime = endTime,
    type = StepType.getById(typeId.toInt()) ?: StepType.Walking
)

```

```

package com.zhiro.jetstep

import android.app.Application
import com.zhiro.jetstep.utils.initmanager.InitManager
import com.zhiro.jetstep.utils.initmanager.KoinInitManager

class App : Application() {

    override fun onCreate() {
        super.onCreate()
        initApp()
    }

    private fun initApp() {
        setOf<InitManager>(
            KoinInitManager(this)
        ).forEach(InitManager::init)
    }
}

<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path
        android:fillColor="#3DDC84"
        android:pathData="M0,0h108v108h-108z" />
    <path
        android:fillColor="#00000000"
        android:pathData="M9,0L9,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,0L19,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M29,0L29,108"
        android:strokeWidth="0.8"

```

```

        android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M39,0L39,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M49,0L49,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M59,0L59,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M69,0L69,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M79,0L79,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M89,0L89,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M99,0L99,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,9L108,9"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path

```

```

        android:fillColor="#00000000"
        android:pathData="M0,19L108,19"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,29L108,29"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,39L108,39"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,49L108,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,59L108,59"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,69L108,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,79L108,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,89L108,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,99L108,99"

```

```

        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,29L89,29"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,39L89,39"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,49L89,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,59L89,59"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,69L89,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,79L89,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M29,19L29,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M39,19L39,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />

```

```

<path
    android:fillColor="#00000000"
    android:pathData="M49,19L49,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M59,19L59,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M69,19L69,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M79,19L79,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
</vector>

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:aapt="http://schemas.android.com/aapt"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4
26,-1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">
        <aapt:attr name="android:fillColor">
            <gradient
                android:endX="85.84757"
                android:endY="92.4963"
                android:startX="42.9492"
                android:startY="49.59793"
                android:type="linear">
                <item
                    android:color="#44000000"
                    android:offset="0.0" />
                <item
                    android:color="#00000000"
                    android:offset="1.0" />
            </gradient>
        </aapt:attr>
    </path>
</vector>

```

```

        </gradient>
    </aapt:attr>
</path>
<path
    android:fillColor="#FFFFFF"
    android:fillType="nonZero"
    android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-
0.4,-0.2 -0.9,-0.1 -1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-
0.2,-0.4 -0.7,-0.5 -1.1,-0.3C38.8,38.328 38.7,38.828
38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028 31,63.928h46C76.3,56.028 71.8,49.428
65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2c-0.3,-0.7 -0.1,-1.5 0.4,-
2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C45.3,56.528 44.5,57.328
43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2s-0.1,-1.5 0.4,-
2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328
64.6,57.328L64.6,57.328z"
    android:strokeWidth="1"
    android:strokeColor="#00000000" />
</vector>

<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@drawable/ic_launcher_background" />
    <foreground android:drawable="@drawable/ic_launcher_foreground" />
    <monochrome android:drawable="@drawable/ic_launcher_foreground" />
</adaptive-icon>

<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@drawable/ic_launcher_background" />
    <foreground android:drawable="@drawable/ic_launcher_foreground" />
    <monochrome android:drawable="@drawable/ic_launcher_foreground" />
</adaptive-icon>

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>

```

```

<resources>
    <string name="fitness_notification_title">Collecting fitness data...</string>
    <string name="fitness_notification_channel_name"
translatable="false">Fitness Notification Channel Name</string>
</resources>

```

```

<?xml version="1.0" encoding="utf-8"?>

```

```

<resources>

```

```

    <style name="Theme.JetStep"
parent="android:Theme.Material.Light.NoActionBar" />
</resources>

```

```

<?xml version="1.0" encoding="utf-8"?>

```

```

<resources>

```

```

    <string name="fitness_notification_title">Збір фітнес-даних...</string>
</resources>

```

```

<?xml version="1.0" encoding="utf-8"?><!--

```

```

    Sample backup rules file; uncomment and customize as necessary.
    See https://developer.android.com/guide/topics/data/autobackup
    for details.

```

```

    Note: This file is ignored for devices older than API 31

```

```

    See https://developer.android.com/about/versions/12/backup-restore

```

```

-->

```

```

<full-backup-content>

```

```

    <!--

```

```

    <include domain="sharedpref" path="."/>

```

```

    <exclude domain="sharedpref" path="device.xml"/>

```

```

-->

```

```

</full-backup-content>

```

```

<?xml version="1.0" encoding="utf-8"?><!--

```

```

    Sample data extraction rules file; uncomment and customize as necessary.

```

```

    See https://developer.android.com/about/versions/12/backup-restore#xml-
changes

```

```

    for details.

```

```

-->

```

```

<data-extraction-rules>

```

```

    <cloud-backup>

```

```

        <!-- TODO: Use <include> and <exclude> to control what is backed up.

```

```

        <include .../>

```

```

        <exclude .../>

```

```

        -->

```

```

</cloud-backup>
<!--
<device-transfer>
    <include .../>
    <exclude .../>
</device-transfer>
-->
</data-extraction-rules>

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission
android:name="android.permission.FOREGROUND_SERVICE_HEALTH"/>
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
    <uses-permission android:name="android.permission.ACTIVITY_RECOGNITION"/>

    <application
        android:name=".App"
        android:allowBackup="false"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.JetStep"
        tools:targetApi="35">

        <activity
            android:name=".android.activities.MainActivity"
            android:exported="true"
            android:theme="@style/Theme.JetStep">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".android.activities.SupportiveActivity"

```

```

        android:exported="true"
        android:theme="@style/Theme.JetStep">
<!--          <intent-filter>-->
<!--          <action android:name="android.intent.action.MAIN" />-->

<!--          <category android:name="android.intent.category.LAUNCHER"
/>-->
<!--          </intent-filter>-->
    </activity>

    <service android:name=".android.services.AccelerometerCollectorService"
/>

    <service
        android:name=".android.services.FitnessForegroundService"
        android:exported="false"
        android:foregroundServiceType="health" />

</application>
</manifest>

plugins {
    alias(libs.plugins.jetstep.jvm.library)
    alias(libs.plugins.jetstep.koin)
}

dependencies {
    implementation(libs.kotlin.coroutines.core)
}

package com.zhiro.jetstep.core.common.coroutines

import kotlinx.coroutines.CoroutineDispatcher
import kotlinx.coroutines.Dispatchers

internal class AndroidAppDispatchers : AppDispatchers {

    override val default: CoroutineDispatcher
        get() = Dispatchers.Default
    override val main: CoroutineDispatcher
        get() = Dispatchers.Main
    override val io: CoroutineDispatcher

```

```

        get() = Dispatchers.IO
    }

package com.zhiro.jetstep.core.common.coroutines

import kotlinx.coroutines.CoroutineDispatcher

interface AppDispatchers {

    val default: CoroutineDispatcher
    val main: CoroutineDispatcher
    val io: CoroutineDispatcher

}

package com.zhiro.jetstep.core.common.coroutines

import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.SupervisorJob
import kotlin.coroutines.CoroutineContext

class ModelCoroutineScope(val dispatchers: AppDispatchers) : CoroutineScope {
    override val coroutineContext: CoroutineContext = SupervisorJob() +
dispatchers.default
}

package com.zhiro.jetstep.core.common.di.module

import com.zhiro.jetstep.core.common.coroutines.AndroidAppDispatchers
import com.zhiro.jetstep.core.common.coroutines.AppDispatchers
import com.zhiro.jetstep.core.common.coroutines.ModelCoroutineScope
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.SupervisorJob
import org.koin.core.definition.BeanDefinition
import org.koin.core.module.dsl.bind
import org.koin.core.module.dsl.factoryOf
import org.koin.dsl.module

val commonModule
    get() = module {
        factoryOf(::AndroidAppDispatchers, BeanDefinition<AppDispatchers>::bind)
        factoryOf(::ModelCoroutineScope)
        factory<CoroutineScope> {

```

```

        CoroutineScope(SupervisorJob() + Dispatchers.Default)
    }
}

package com.zhiro.jetstep.core.common.utils

import java.time.LocalDate
import java.time.LocalDateTime

fun LocalDate.atEndOfDay(): LocalDateTime {
    return plusDays(1).atStartOfDay().minusSeconds(1)
}

<?xml version="1.0" encoding="utf-8"?>
<manifest>

</manifest>

plugins {
    alias(libs.plugins.jetstep.android.library)
    alias(libs.plugins.jetstep.android.library.compose)
}

android {
    namespace = "com.zhiro.jetstep.core.design.system"
}

dependencies {}

package com.zhiro.jetstep.core.design.system.theme

import androidx.compose.ui.graphics.Color

val Purple80 = Color(color = 0xFFD0BCFF)
val PurpleGrey80 = Color(color = 0xFFCCC2DC)
val Pink80 = Color(color = 0xFFE8B88C)

val Purple40 = Color(color = 0xFF6650a4)
val PurpleGrey40 = Color(color = 0xFF625b71)
val Pink40 = Color(color = 0xFF7D5260)

val Color.Companion.Jogging get() = Purple80
val Color.Companion.Running get() = PurpleGrey80
val Color.Companion.Walking get() = Pink80

```

```

package com.zhiro.jetstep.core.design.system.theme

import android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.dynamicDarkColorScheme
import androidx.compose.material3.dynamicLightColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.ui.platform.LocalContext

private val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)

private val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = PurpleGrey40,
    tertiary = Pink40
)

@Composable
fun JetStepTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context) else
dynamicLightColorScheme(context)
        }

        darkTheme -> DarkColorScheme
        else -> LightColorScheme
    }
}

```

```

    MaterialTheme(
        colorScheme = colorScheme,
        typography = Typography,
        content = content
    )
}

package com.zhiro.jetstep.core.design.system.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    bodyLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp,
        lineHeight = 24.sp,
        letterSpacing = 0.5.sp
    )
)

<?xml version="1.0" encoding="utf-8"?>
<manifest>

</manifest>

plugins {
    alias(libs.plugins.jetstep.android.library)
    alias(libs.plugins.jetstep.android.library.compose)
    alias(libs.plugins.jetstep.android.koin)
}

android {
    namespace = "com.zhiro.jetstep.core.navigation"
}

dependencies {

    implementation(projects.core.ui)

```

```

        api(libs.androidx.navigation.compose)
    }

package com.zhiro.jetstep.core.navigation.di.modules

import com.zhiro.jetstep.core.navigation.domain.NavigationManager
import com.zhiro.jetstep.core.navigation.domain.NavigationManagerImpl
import org.koin.dsl.module

val navigationModule
    get() = module {
        single<NavigationManager> {
            NavigationManagerImpl()
        }
    }

package com.zhiro.jetstep.core.navigation.domain

import androidx.activity.ComponentActivity
import androidx.navigation.NavHostController
import androidx.navigation.NavOptionsBuilder

interface NavigationManager {

    fun <T : Any> navigateTo(route: T, builder: NavOptionsBuilder.() -> Unit =
    {})

    fun popBackStack()

    /**
     * Should be called on Activity.onCreate()
     */
    fun attach(activity: ComponentActivity, navController: NavHostController)

    /**
     * Should be called on Activity.onDestroy()
     */
    fun detach()
}

class NavigationManagerImpl : NavigationManager {

    private var navController: NavHostController? = null

```

```

private var activity: ComponentActivity? = null

override fun <T : Any> navigateTo(route: T, builder: NavOptionsBuilder.() ->
Unit) {
    navController?.navigate(route = route, builder = builder)
}

override fun popBackStack() {
    if (navController?.navigateUp() != true) {
        activity?.onBackPressedDispatcher?.onBackPressed()
    }
}

/**
 * Should be called on Activity.onCreate()
 */
override fun attach(activity: ComponentActivity, navController:
NavHostController) {
    this.activity = activity
    this.navController = navController
}

/**
 * Should be called on Activity.onDestroy()
 */
override fun detach() {
    this.navController = null
    this.activity = null
}
}

package com.zhiro.jetstep.core.navigation.routes

import kotlinx.serialization.Serializable

sealed interface ScreenRoute {

    @Serializable
    data object Dashboard : ScreenRoute
}

package com.zhiro.jetstep.core.navigation.utils

```

```

import androidx.activity.ComponentActivity
import androidx.compose.runtime.Composable
import androidx.compose.runtime.DisposableEffect
import androidx.compose.runtime.remember
import androidx.navigation.NavHostController
import com.zhiro.jetstep.core.navigation.domain.NavigationManager
import com.zhiro.jetstep.core.ui.localcomposition.LocalActivity
import org.koin.core.context.GlobalContext

private fun attachNavController(navController: NavHostController, activity:
ComponentActivity) {
    getNavigationManager().attach(activity = activity, navController =
navController)
}

private fun detachNavController() {
    getNavigationManager().detach()
}

@Composable
fun rememberNavigationManager(): NavigationManager {
    return remember {
        getNavigationManager()
    }
}

private fun getNavigationManager(): NavigationManager {
    return GlobalContext.get().get<NavigationManager>()
}

@Composable
fun InitNavigationManager(navController: NavHostController) {
    val activity = LocalActivity.current

    if (activity != null) {
        DisposableEffect(Unit) {
            attachNavController(activity = activity, navController =
navController)

            onDispose {
                detachNavController()
            }
        }
    }
}

```

```

    }
}
}

<?xml version="1.0" encoding="utf-8"?>
<manifest>

</manifest>

package com.zhiro.jetstep.core.ui.base.mvi.util

import java.util.UUID

data class BaseError(
    val message: String?,
    val key: String = UUID.randomUUID().toString()
)

fun Throwable.toError() = BaseError(message = message)

package com.zhiro.jetstep.core.ui.base.mvi.util

import com.zhiro.jetstep.core.ui.base.mvi.BaseViewModel
import kotlinx.coroutines.ExperimentalCoroutinesApi
import kotlinx.coroutines.FlowPreview
import kotlinx.coroutines.flow.MutableStateFlow
import kotlin.properties.ReadWriteProperty
import kotlin.reflect.KProperty

@OptIn(FlowPreview::class, ExperimentalCoroutinesApi::class)
fun <T> MutableStateFlow<T>.delegate() = object :
ReadWriteProperty<BaseViewModel<*, *, *>, T> {

    override fun setValue(
        thisRef: BaseViewModel<*, *, *>,
        property: KProperty<*>,
        value: T
    ) {
        this@delegate.value = value
    }

    override fun getValue(thisRef: BaseViewModel<*, *, *>, property:
KProperty<*>): T {
        return requireNotNull(this@delegate.value)
    }
}

```

```

    }
}

package com.zhiro.jetstep.core.ui.base.mvi

import androidx.lifecycle.ViewModel
import com.zhiro.jetstep.core.common.coroutines.AppDispatchers
import com.zhiro.jetstep.core.common.coroutines.ModelCoroutineScope
import com.zhiro.jetstep.core.ui.base.mvi.util.delegate
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.ExperimentalCoroutinesApi
import kotlinx.coroutines.FlowPreview
import kotlinx.coroutines.cancel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.collectLatest
import kotlinx.coroutines.flow.flattenMerge
import kotlinx.coroutines.flow.flowOf
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext

@FlowPreview
@ExperimentalCoroutinesApi
abstract class BaseViewModel<Action : MviAction<State>, State : Any, UiState :
Any>(
    private val reducer: Reducer<Action, State>,
    private val stateMapper: StateMapper<State, UiState>,
    private val useCase: Set<UseCase<Action, State, out Action>>,
    private val coroutineScope: ModelCoroutineScope,
    initialAction: Action,
) : ViewModel(), CoroutineScope by coroutineScope {

    private val mutableState = MutableStateFlow(reducer.initialState)
    protected var stateValue: State by mutableState.delegate()
    val state: StateFlow<UiState> = createUiStateFlow()

    private val actionListeners =
mutableListOf<ViewModelActionHandler<Action>>()
    protected val dispatchers: AppDispatchers
        get() = coroutineScope.dispatchers

    init {
        launch(dispatchers.main) {

```

```

        flowOf(
            *useCase.filterIsInstance<SideEffectUseCase<Action, State, *>>()
                .onEach {
                    it.stateProvider = { stateValue }
                }.map {
                    it.sideEffectFlow
                }.toArray()
        ).flattenMerge()
            .collect {
                action(it)
            }
        }
        action(initialAction)
    }

protected fun action(action: Action) {
    stateValue = reducer.reduce(action, stateValue)
    notifyListeners(action)
    filterUseCase(action).forEach {
        launch(it.dispatcher) {
            val result = it.invoke(action, stateValue)
            withContext(dispatchers.main) {
                action(result)
            }
        }
    }
}

@Suppress("UNCHECKED_CAST")
private fun <T : Action> filterUseCase(action: T): List<UseCase<Action,
State, T>> {
    return useCase.mapNotNull {
        it as? UseCase<Action, State, T>
    }.filter {
        it.canHandle(action)
    }
}

private fun notifyListeners(action: Action) {
    actionListeners.filter {
        it.filter(action)
    }.forEach {

```

```

        it.onAction(action)
    }
}

protected fun handleAction(filter: (Action) -> Boolean, onAction: (Action) -
> Unit) {
    actionListeners.add(ViewModelActionHandler(filter, onAction))
}

private fun createUiStateFlow(): StateFlow<UiState> {
    val uiStateFlow =
MutableStateFlow(stateMapper.toUi(reducer.initialState))
    launch(dispatchers.main) {
        mutableState.collectLatest {
            uiStateFlow.value = stateMapper.toUi(it)
        }
    }
    return uiStateFlow
}

override fun onCleared() {
    cancel()
    super.onCleared()
}
}

package com.zhiro.jetstep.core.ui.base.mvi

interface MviAction<State> {

    fun accept(state: State): State {
        return state
    }
}

package com.zhiro.jetstep.core.ui.base.mvi

interface Reducer<Action : MviAction<State>, State> {

    val initialState: State

    fun reduce(action: Action, state: State): State {
        return action.accept(state)
    }
}

```

```

    }
}

package com.zhiro.jetstep.core.ui.base.mvi

fun interface StateMapper<State : Any, UiState : Any> {
    fun toUi(state: State): UiState
}

package com.zhiro.jetstep.core.ui.base.mvi

import com.zhiro.jetstep.core.common.coroutines.AppDispatchers
import kotlinx.coroutines.CoroutineDispatcher
import kotlinx.coroutines.flow.Flow
import java.lang.reflect.ParameterizedType

abstract class UseCase<Action, State, A : Action>(protected open val
dispatchers: AppDispatchers) {
    open val dispatcher: CoroutineDispatcher = dispatchers.io
    private val actionClass by lazy {
        (javaClass.genericSuperclass as
ParameterizedType).actualTypeArguments[2]
    }

    abstract suspend operator fun invoke(action: A, state: State): Action

    open fun canHandle(action: Action): Boolean {
        return action!!::class.java == actionClass
    }
}

abstract class SideEffectUseCase<Action, State, A : Action>(
    dispatchers: AppDispatchers
) : UseCase<Action, State, A>(dispatchers) {
    lateinit var stateProvider: () -> State
    abstract val sideEffectFlow: Flow<Action>
}

interface GenericUseCase<T> {
    suspend fun <Action> invoke(
        onSuccess: suspend (T) -> Action,
        onError: suspend (Exception) -> Action
    ): Action
}

```

```

}

interface ErrorFlowComponent {
    val errorFlow: Flow<String>
}

package com.zhiro.jetstep.core.ui.base.mvi

class ViewModelActionHandler<Action>(
    val filter: (Action) -> Boolean,
    val onAction: (Action) -> Unit
)

package com.zhiro.jetstep.core.ui.component.spacer

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.navigationBarsPadding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.statusBarsPadding
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.Dp

@Composable
fun SimpleSpacer(space: Dp) {
    Spacer(modifier = Modifier.size(size = space))
}

@Composable
fun SystemBarSpacer() {
    Spacer(
        modifier = Modifier
            .statusBarsPadding()
            .background(Color.Green)
    )
}

@Composable
fun NavigationBarSpacer() {
    Spacer(
        modifier = Modifier

```

```

        .navigationBarPadding()
        .background(Color.Green)
    )
}

package com.zhiro.jetstep.core.ui.localcomposition

import android.content.Context
import android.content.ContextWrapper
import androidx.activity.ComponentActivity
import androidx.compose.runtime.Composable
import androidx.compose.ui.platform.LocalContext

val LocalActivity = object : MockCompositionLocal<ComponentActivity?> {

    override val current: ComponentActivity?
        @Composable
        get() = LocalContext.current.activity
}

@Suppress("RecursivePropertyAccessor")
val Context.activity: ComponentActivity?
    get() = when (this) {
        is ComponentActivity -> this
        is ContextWrapper -> baseContext.activity
        else -> null
    }

package com.zhiro.jetstep.core.ui.localcomposition

import androidx.compose.runtime.Composable
import androidx.compose.runtime.Stable

@Stable
interface MockCompositionLocal<R> {

    @get:Composable
    val current: R
}

package com.zhiro.jetstep.core.ui.utils

import androidx.compose.runtime.Composable
import androidx.compose.ui.graphics.Color

```

```

import androidx.compose.ui.graphics.painter.Painter
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import com.zhiro.jetstep.core.design.system.theme.Jogging
import com.zhiro.jetstep.core.design.system.theme.Running
import com.zhiro.jetstep.core.design.system.theme.Walking
import com.zhiro.jetstep.core.ui.R
import com.zhiro.jetstep.domain.model.StepType

val StepType.color: Color
    get() = when (this) {
        StepType.Walking -> Color.Walking
        StepType.Jogging -> Color.Jogging
        StepType.Running -> Color.Running
    }

val StepType.icon: Painter
    @Composable
    get() = when (this) {
        StepType.Walking -> painterResource(R.drawable.ic_walking)
        StepType.Jogging -> painterResource(R.drawable.ic_jogging)
        StepType.Running -> painterResource(R.drawable.ic_running)
    }

val StepType.text: String
    @Composable
    get() = when (this) {
        StepType.Walking -> stringResource(R.string.walking)
        StepType.Jogging -> stringResource(R.string.jogging)
        StepType.Running -> stringResource(R.string.running)
    }

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="960"
    android:viewportHeight="960"
    android:tint="?attr/colorControlNormal"
    android:autoMirrored="true">
    <path
        android:fillColor="@android:color/white"
        android:pathData="M520,880L520,680L436,600L405,738Q401,754
387.5,762.5Q374,771 358,768L160,728Q143,725 134,711Q125,697 128,680Q131,663
145,653.5Q159,644 176,648L328,680L392,356L320,384L320,480Q320,497

```

```

308.5,508.5Q297,520 280,520Q263,520 251.5,508.5Q240,497 240,480L240,358Q240,346
246.5,336.5Q253,327 264,322L398,264Q433,249 449.5,244.5Q466,240 480,240Q501,240
519,251Q537,262 548,280L588,344Q609,378 642.5,403Q676,428 720,436Q737,439
748.5,451Q760,463 760,480Q760,497 748.5,508Q737,519 721,517Q667,509
620,483.5Q573,458 540,420L516,540L588,608Q594,614 597,621.5Q600,629
600,637L600,880Q600,897 588.5,908.5Q577,920 560,920Q543,920 531.5,908.5Q520,897
520,880ZM540,220Q507,220 483.5,196.5Q460,173 460,140Q460,107 483.5,83.5Q507,60
540,60Q573,60 596.5,83.5Q620,107 620,140Q620,173 596.5,196.5Q573,220 540,220Z"/>
</vector>

```

```

<vector xmlns:android="http://schemas.android.com/apk/res/android"

```

```

    android:width="24dp"

```

```

    android:height="24dp"

```

```

    android:viewportWidth="960"

```

```

    android:viewportHeight="960"

```

```

    android:tint="?attr/colorControlNormal">

```

```

<path

```

```

    android:fillColor="@android:color/white"

```

```

    android:pathData="M482,535L245,772Q233,784 216.5,784Q200,784
188,772Q176,760 176,743.5Q176,727 188,715L544,360L440,360L440,400Q440,417
428.5,428.5Q417,440 400,440Q383,440 371.5,428.5Q360,417 360,400L360,320Q360,303
371.5,291.5Q383,280 400,280L594,280Q610,280 624.5,286Q639,292
650,303L770,422Q790,442 816.5,455.5Q843,469 874,475Q891,478 903.5,490.5Q916,503
916,520Q916,537 902,548Q888,559 871,557Q826,551 788,533Q750,515
718,484L678,442L590,530L643,583Q657,597 654.5,616.5Q652,636
635,646L453,751Q439,759 422.5,755Q406,751 398,737Q390,723 394.5,706.5Q399,690
413,682L550,603L482,535ZM160,520Q143,520 131.5,508.5Q120,497 120,480Q120,463
131.5,451.5Q143,440 160,440L280,440Q297,440 308.5,451.5Q320,463 320,480Q320,497
308.5,508.5Q297,520 280,520L160,520ZM80,400Q63,400 51.5,388.5Q40,377
40,360Q40,343 51.5,331.5Q63,320 80,320L200,320Q217,320 228.5,331.5Q240,343
240,360Q240,377 228.5,388.5Q217,400 200,400L80,400ZM779,320Q746,320
722,296.5Q698,273 698,240Q698,207 722,183.5Q746,160 779,160Q812,160
836,183.5Q860,207 860,240Q860,273 836,296.5Q812,320 779,320ZM160,280Q143,280
131.5,268.5Q120,257 120,240Q120,223 131.5,211.5Q143,200 160,200L280,200Q297,200
308.5,211.5Q320,223 320,240Q320,257 308.5,268.5Q297,280 280,280L160,280Z"/>

```

```

</vector>

```

```

<vector xmlns:android="http://schemas.android.com/apk/res/android"

```

```

    android:width="24dp"

```

```

        android:height="24dp"
        android:viewportWidth="960"
        android:viewportHeight="960"
        android:tint="?attr/colorControlNormal"
        android:autoMirrored="true">
<path
    android:fillColor="@android:color/white"
    android:pathData="M436,596L371,888Q368,902 356.5,911Q345,920
330,920Q310,920 298,905Q286,890 290,871L392,356L320,384L320,480Q320,497
308.5,508.5Q297,520 280,520Q263,520 251.5,508.5Q240,497 240,480L240,358Q240,346
246.5,336.5Q253,327 264,322L442,246Q456,240 471.5,239Q487,238 501,243Q515,248
527.5,257Q540,266 548,280L588,344Q601,364 618.5,382Q636,400 658,413Q672,421
689,427.5Q706,434 723,437Q739,440 749.5,451.5Q760,463 760,480Q760,497
748,508Q736,519 719,517Q663,509 618.5,482Q574,455
541,417L516,540L588,608Q594,614 597,621.5Q600,629 600,637L600,880Q600,897
588.5,908.5Q577,920 560,920Q543,920 531.5,908.5Q520,897
520,880L520,660L436,596ZM540,220Q507,220 483.5,196.5Q460,173 460,140Q460,107
483.5,83.5Q507,60 540,60Q573,60 596.5,83.5Q620,107 620,140Q620,173
596.5,196.5Q573,220 540,220Z"/>
</vector>

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="running">Running</string>
    <string name="jogging">Jogging</string>
    <string name="walking">Walking</string>
</resources>

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="running">Біг</string>
    <string name="jogging">Біг підтюпцем</string>
    <string name="walking">Ходьба</string>
</resources>

<?xml version="1.0" encoding="utf-8"?>
<manifest>

</manifest>

plugins {
    alias(libs.plugins.jetstep.android.library)
    alias(libs.plugins.jetstep.room.library)
    alias(libs.plugins.jetstep.koin)

```

```

}

android {
    namespace = "com.zhiro.jetstep.data"
}

dependencies {
    implementation(libs.androidx.datastore.preferences)
}

package com.zhiro.jetstep.datasource.room.base

import androidx.room.Delete
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Update

interface BaseDao<in T> {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insert(entity: T)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insert(entities: List<T>)

    @Update
    fun update(entity: T)

    @Update
    fun update(entities: List<T>)

    @Delete
    fun delete(entity: T)

    @Delete
    fun delete(entities: List<T>)
}

package com.zhiro.jetstep.datasource.room.dao

import androidx.room.Dao
import androidx.room.Query
import com.zhiro.jetstep.datasource.room.base.BaseDao
import com.zhiro.jetstep.datasource.room.models.AccelerometerEntity

```

```

import kotlinx.coroutines.flow.Flow

@Dao
interface AccelerometerDao : BaseDao<AccelerometerEntity> {

    @Query("SELECT * FROM ${AccelerometerEntity.TABLE_NAME}")
    fun readAll(): Flow<List<AccelerometerEntity>>

    @Query("DELETE FROM ${AccelerometerEntity.TABLE_NAME}")
    fun dropTable()
}

package com.zhiro.jetstep.datasource.room.dao

import androidx.room.Dao
import androidx.room.Query
import com.zhiro.jetstep.datasource.room.base.BaseDao
import com.zhiro.jetstep.datasource.room.models.AccelerometerEntity
import com.zhiro.jetstep.datasource.room.models.StepBatchEntity
import kotlinx.coroutines.flow.Flow

@Dao
interface StepBatchDao : BaseDao<StepBatchEntity> {

    @Query("SELECT * FROM ${StepBatchEntity.TABLE_NAME}")
    fun readAll(): Flow<List<StepBatchEntity>>

    @Query("SELECT * FROM ${StepBatchEntity.TABLE_NAME} WHERE
${StepBatchEntity.COLUMN_START_TIME} BETWEEN :startTime AND :endTime")
    fun readInterval(startTime: Long, endTime: Long):
Flow<List<StepBatchEntity>>

    @Query("DELETE FROM ${AccelerometerEntity.TABLE_NAME}")
    fun dropTable()
}

package com.zhiro.jetstep.datasource.room.database

import androidx.room.Database
import androidx.room.RoomDatabase
import com.zhiro.jetstep.datasource.room.dao.AccelerometerDao
import com.zhiro.jetstep.datasource.room.dao.StepBatchDao
import com.zhiro.jetstep.datasource.room.models.AccelerometerEntity

```

```

import com.zhiro.jetstep.datasource.room.models.StepBatchEntity

@Database(
    entities = [AccelerometerEntity::class, StepBatchEntity::class],
    version = SensorDatabase.DATABASE_VERSION,
    exportSchema = true,
)
internal abstract class SensorDatabase : RoomDatabase() {

    abstract fun getAccelerometerDao(): AccelerometerDao

    abstract fun getStepBatchDao(): StepBatchDao

    companion object {

        internal const val DATABASE_NAME = "fitness_database"
        internal const val DATABASE_VERSION = 1
    }
}

package com.zhiro.jetstep.datasource.room.models

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = AccelerometerEntity.TABLE_NAME)
data class AccelerometerEntity(
    @PrimaryKey
    @ColumnInfo(name = COLUMN_TIMESTAMP) val timestamp: Long,
    @ColumnInfo(name = COLUMN_X) val x: Float,
    @ColumnInfo(name = COLUMN_Y) val z: Float,
    @ColumnInfo(name = COLUMN_Z) val y: Float,
    @ColumnInfo(name = COLUMN_PACE) val pace: String,
) {

    internal companion object {

        const val TABLE_NAME = "accelerometer_data"

        const val COLUMN_X = "x"
        const val COLUMN_Y = "y"
        const val COLUMN_Z = "z"
    }
}

```

```

        const val COLUMN_TIMESTAMP = "timestamp"
        const val COLUMN_PACE = "pace"
    }
}

package com.zhiro.jetstep.datasource.room.models

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
import
com.zhiro.jetstep.datasource.room.models.StepBatchEntity.Companion.TABLE_NAME

@Entity(tableName = TABLE_NAME)
data class StepBatchEntity(
    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = COLUMN_ID) val id: Int? = null,
    @ColumnInfo(name = COLUMN_STEPS) val steps: Int,
    @ColumnInfo(name = COLUMN_TYPE_ID) val typeId: Int,
    @ColumnInfo(name = COLUMN_START_TIME) val startTime: Long,
    @ColumnInfo(name = COLUMN_END_TIME) val endTime: Long,
) {

    internal companion object {

        const val TABLE_NAME = "step_batch_data"

        const val COLUMN_ID = "id"
        const val COLUMN_STEPS = "steps"
        const val COLUMN_START_TIME = "start_time"
        const val COLUMN_END_TIME = "end_time"
        const val COLUMN_TYPE_ID = "type_id"
    }
}

plugins {
    alias(libs.plugins.jetstep.android.library)
    alias(libs.plugins.jetstep.koin)
}

android {
    namespace = "com.zhiro.jetstep.domain"
}

```

```

dependencies {
    implementation(projects.data)
    api(projects.domainModel)

    implementation(libs.kotlin.coroutines.core)
}

package com.zhiro.jetstep.domain.di.module

import com.zhiro.jetstep.datasource.di.module.dataModule
import com.zhiro.jetstep.domain.repository.SensorRepository
import com.zhiro.jetstep.domain.repository.SensorRepositoryImpl
import com.zhiro.jetstep.domain.repository.StepRepository
import com.zhiro.jetstep.domain.repository.StepRepositoryImpl
import org.koin.core.definition.BeanDefinition
import org.koin.core.module.dsl.bind
import org.koin.core.module.dsl.factoryOf
import org.koin.dsl.module

val domainModule
    get() = module {
        includes(dataModule)

        // Repository set up
        factoryOf(::SensorRepositoryImpl,
            BeanDefinition<SensorRepository>::bind)
        factoryOf(::StepRepositoryImpl, BeanDefinition<StepRepository>::bind)
    }

package com.zhiro.jetstep.domain.repository

import com.zhiro.jetstep.datasource.room.dao.AccelerometerDao
import com.zhiro.jetstep.datasource.room.models.AccelerometerEntity
import com.zhiro.jetstep.domain.model.RawAccelerometerRecord
import com.zhiro.jetstep.domain.model.toDomain
import com.zhiro.jetstep.domain.model.toEntity
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.firstOrNull
import kotlinx.coroutines.flow.map

interface SensorRepository {

```

```

        suspend fun saveRawAccelerometerRecords(records:
List<RawAccelerometerRecord>)

        suspend fun getAllRawAccelerometerRecords(): List<RawAccelerometerRecord>

        fun getAllRawAccelerometerRecordsFlow(): Flow<List<RawAccelerometerRecord>>

        fun deleteAllData()
    }

class SensorRepositoryImpl(
    private val accelerometerDao: AccelerometerDao,
) : SensorRepository {

    override suspend fun saveRawAccelerometerRecords(records:
List<RawAccelerometerRecord>) {
        accelerometerDao.insert(records.map(RawAccelerometerRecord::toEntity))
    }

    override suspend fun getAllRawAccelerometerRecords():
List<RawAccelerometerRecord> {
        return
        accelerometerDao.readAll().firstOrNull()?.map(AccelerometerEntity::toDomain)
            ?: emptyList()
    }

    override fun getAllRawAccelerometerRecordsFlow():
Flow<List<RawAccelerometerRecord>> {
        return accelerometerDao.readAll().map { accelerometerRecords ->
            accelerometerRecords.map(AccelerometerEntity::toDomain)
        }
    }

    override fun deleteAllData() {
        accelerometerDao.dropTable()
    }
}

package com.zhiro.jetstep.domain.repository

import com.zhiro.jetstep.datasource.room.dao.StepBatchDao
import com.zhiro.jetstep.datasource.room.models.StepBatchEntity
import com.zhiro.jetstep.domain.model.StepBatch

```

```

import com.zhiro.jetstep.domain.model.toData
import com.zhiro.jetstep.domain.model.toDomain
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.map

interface StepRepository {

    fun loadStepBatches(startTime: Long, endTime: Long): Flow<List<StepBatch>>

    suspend fun saveStepBatches(stepBatches: List<StepBatch>)

    suspend fun saveStepBatch(stepBatch: StepBatch)
}

class StepRepositoryImpl(private val stepBatchDao: StepBatchDao) :
    StepRepository {

    override fun loadStepBatches(startTime: Long, endTime: Long):
        Flow<List<StepBatch>> {
        return stepBatchDao.readInterval(startTime = startTime, endTime =
        endTime).map { entities ->
            entities.map(StepBatchEntity::toDomain)
        }
    }

    override suspend fun saveStepBatches(stepBatches: List<StepBatch>) {
        stepBatchDao.insert(stepBatches.map(StepBatch::toData))
    }

    override suspend fun saveStepBatch(stepBatch: StepBatch) {
        stepBatchDao.insert(stepBatch.toData())
    }
}

plugins {
    alias(libs.plugins.jetstep.android.library)
}

android {
    namespace = "com.zhiro.jetstep.domain.model"
}

dependencies {

```

```

        implementation(projects.data)

        implementation(libs.kotlin.coroutines.core)
    }

package com.zhiro.jetstep.domain.model

import com.zhiro.jetstep.datasource.room.models.AccelerometerEntity

class RawAccelerometerRecord(
    val x: Float,
    val y: Float,
    val z: Float,
    val recordTakenAt: Long,
    val pace: String,
)

fun RawAccelerometerRecord.toEntity(): AccelerometerEntity {
    return AccelerometerEntity(
        x = x,
        y = y,
        z = z,
        timestamp = recordTakenAt,
        pace = pace
    )
}

fun AccelerometerEntity.toDomain(): RawAccelerometerRecord {
    return RawAccelerometerRecord(
        x = x,
        y = y,
        z = z,
        recordTakenAt = timestamp,
        pace = pace
    )
}

package com.zhiro.jetstep.domain.model

import com.zhiro.jetstep.datasource.room.models.StepBatchEntity

data class StepBatch(
    val steps: Int,

```

```

        val startTime: Long,
        val endTime: Long,
        val type: StepType
    )

    val StepBatch.duration get() = endTime - startTime

    fun List<StepBatch>.aggregate(): Map<StepType, Int> {
        var walkingSteps = 0
        var joggingSteps = 0
        var runningSteps = 0

        forEach { batch ->
            when (batch.type) {
                StepType.Walking -> {
                    walkingSteps += batch.steps
                }
                StepType.Jogging -> {
                    joggingSteps += batch.steps
                }
                StepType.Running -> {
                    runningSteps += batch.steps
                }
            }
        }

        return mapOf(
            StepType.Walking to walkingSteps,
            StepType.Jogging to joggingSteps,
            StepType.Running to runningSteps,
        )
    }

    fun StepBatchEntity.toDomain() = StepBatch(
        steps = steps,
        startTime = startTime,
        endTime = endTime,
        type = StepType.getById(id = typeId) ?: StepType.Walking
    )

    fun StepBatch.toData() = StepBatchEntity(
        steps = steps,

```

```

        startTime = startTime,
        endTime = endTime,
        typeId = type.id
    )
}

package com.zhiro.jetstep.domain.model

enum class StepType(val id: Int) {
    Walking(id = 0),
    Jogging(id = 1),
    Running(id = 2);

    companion object {

        fun getById(id: Int?) = entries.find { type -> type.id == id }
    }
}

fun StepType.next(): StepType {
    val nextIndex = if (ordinal + 1 == StepType.entries.size) {
        0
    } else {
        ordinal + 1
    }
    return StepType.entries[nextIndex]
}

plugins {
    alias(libs.plugins.jetstep.android.library)
}

android {
    namespace = "com.zhiro.jetstep.domain.test"
}

dependencies {
    implementation(projects.domain)
    implementation(libs.kotlin.coroutines.core)
}

package com.zhiro.jetstep.domain.test.model

import com.zhiro.jetstep.domain.model.StepBatch
import com.zhiro.jetstep.domain.model.StepType

```

```

private const val MINUTE = 60 * 1000
private const val HOUR = 60 * 60 * 1000
val currentUnixTime = System.currentTimeMillis() // Get current Unix time

val fakeStepBatches1 = run {

    listOf(
        StepBatch(
            steps = 500,
            startTime = currentUnixTime, // Start from now
            endTime = currentUnixTime + 10 * MINUTE, // 10 minutes later
            type = StepType.Walking
        ),
        StepBatch(
            steps = 750,
            startTime = currentUnixTime + 20 * MINUTE, // 20 minutes later
            endTime = currentUnixTime + 30 * MINUTE, // 30 minutes later
            type = StepType.Running
        ),
        StepBatch(
            steps = 200,
            startTime = currentUnixTime + 40 * MINUTE, // 40 minutes later
            endTime = currentUnixTime + 50 * MINUTE, // 50 minutes later
            type = StepType.Jogging
        ),
        StepBatch(
            steps = 1000,
            startTime = currentUnixTime + 60 * MINUTE, // 1 hour later
            endTime = currentUnixTime + 10 * MINUTE + HOUR, // 1 hour and 10
minutes later
            type = StepType.Walking
        ),
        StepBatch(
            steps = 300,
            startTime = currentUnixTime + 20 * MINUTE + HOUR, // 1 hour 20
minutes later
            endTime = currentUnixTime + 30 * MINUTE + HOUR, // 1 hour 30
minutes later
            type = StepType.Walking
        )
    )
}

```

```

}

val fakeStepBatches2 = run {

    listOf(
        StepBatch(
            steps = 1000,
            startTime = currentUnixTime + 30 * MINUTE + HOUR, // 1 hour 20
minutes later
            endTime = currentUnixTime + 50 * MINUTE + HOUR,    // 1 hour 30
minutes later
            type = StepType.Running
        ),
        StepBatch(
            steps = 500,
            startTime = currentUnixTime + 50 * MINUTE + HOUR, // 1 hour 20
minutes later
            endTime = currentUnixTime + 10 * MINUTE + 2 * HOUR,    // 1 hour 30
minutes later
            type = StepType.Jogging
        )
    )
}

package com.zhiro.jetstep.domain.test.repository

import com.zhiro.jetstep.domain.model.StepBatch
import com.zhiro.jetstep.domain.repository.StepRepository
import com.zhiro.jetstep.domain.test.model.fakeStepBatches1
import com.zhiro.jetstep.domain.test.model.fakeStepBatches2
import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow

class FakeStepRepositoryImpl : StepRepository {

    override suspend fun loadStepBatches(startTime: Long, endTime: Long):
Flow<List<StepBatch>> {
        return flow {
            emit(fakeStepBatches1)
            delay(7_000)
            emit(fakeStepBatches2)
        }
    }
}

```

```

    }

    override suspend fun saveStepBatches(stepBatches: List<StepBatch>) {

    }
}

plugins {
    alias(libs.plugins.jetstep.android.feature)
}

android {
    namespace = "com.zhiro.jetstep.feature.dashboard"
}

dependencies {
    implementation(projects.feature.locomotionclassifier)
}

package com.zhiro.jetstep.feature.dashboard.di.module

import org.koin.dsl.module

val featureDashboardModule
    get() = module {
        includes(useCaseModule, viewModelModule)
    }

package com.zhiro.jetstep.feature.dashboard.di.module

import
com.zhiro.jetstep.feature.dashboard.presentation.dashboard.usecase.LoadStepBatch
esUseCase
import
com.zhiro.jetstep.feature.dashboard.presentation.dashboard.usecase.SaveStepBatch
ManuallyUseCase
import org.koin.core.module.dsl.factoryOf
import org.koin.dsl.module

internal val useCaseModule
    get() = module {
        factoryOf(::LoadStepBatchesUseCase)
        factoryOf(::SaveStepBatchManuallyUseCase)
    }

```

```
package com.zhiro.jetstep.feature.dashboard.di.module
```

```
import
com.zhiro.jetstep.feature.dashboard.presentation.dashboard.DashboardViewModel
import org.koin.core.module.dsl.viewModelOf
import org.koin.dsl.module
```

```
internal val viewModelModule
    get() = module {
        viewModelOf(::DashboardViewModel)
    }
```

```
package com.zhiro.jetstep.feature.dashboard.presentation.dashboard.components
```

```
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.wrapContentHeight
import androidx.compose.material3.Button
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.ModalBottomSheet
import androidx.compose.material3.Text
import androidx.compose.material3.rememberModalBottomSheetState
import androidx.compose.runtime.Composable
import androidx.compose.runtime.derivedStateOf
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.rememberCoroutineScope
import androidx.compose.runtime.setValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.zhiro.jetstep.domain.model.StepBatch
import com.zhiro.jetstep.domain.model.StepType
import com.zhiro.jetstep.feature.dashboard.R
import kotlinx.coroutines.launch
import java.text.SimpleDateFormat
import java.util.Date
```

```
import java.util.Locale
```

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
internal fun AddStepsModal(
    modifier: Modifier = Modifier,
    onDismiss: () -> Unit = {},
    onSave: (StepBatch) -> Unit = {},
) {
    val bottomSheetState = rememberModalBottomSheetState(skipPartiallyExpanded =
true)
    val scope = rememberCoroutineScope()
    var startTimeStamp: Long? by remember { mutableStateOf(null) }
    var endTimeStamp: Long? by remember { mutableStateOf(null) }
    val startDateTime by remember(startTimeStamp) {
        val sdf = SimpleDateFormat(TIME_PATTERN, Locale.getDefault())
        mutableStateOf(
            startTimeStamp?.let { time ->
                sdf.format(Date(time))
            } ?: ""
        )
    }
    val endDateTime by remember(endTimeStamp) {
        val sdf = SimpleDateFormat(TIME_PATTERN, Locale.getDefault())
        mutableStateOf(
            endTimeStamp?.let { time ->
                sdf.format(Date(time))
            } ?: ""
        )
    }
    var stepType by remember {
        mutableStateOf(StepType.Walking)
    }
    var stepsValue by remember {
        mutableStateOf("")
    }
    val ctaEnabled by remember(startTimeStamp, endTimeStamp, stepsValue) {
        derivedStateOf {
            startTimeStamp != null && endTimeStamp != null &&
stepsValue.isNotEmpty()
        }
    }
```

```

}

ModalBottomSheet(
    modifier = modifier
        .wrapContentHeight(),
    onDismissRequest = {
        scope.launch {
            bottomSheetState.hide()
            onDismiss()
        }
    },
    sheetState = bottomSheetState
) {

    Column(
        modifier = Modifier.padding(bottom = 24.dp),
        verticalArrangement = Arrangement.spacedBy(16.dp)
    ) {

        DateTimeOutlinedField(
            modifier = Modifier.padding(horizontal = 16.dp),
            value = startDateTime,
            hint = stringResource(R.string.pick_start_time_hint),
            onValueChange = { timestamp ->
                startTimeStamp = timestamp
            },
            isError = false
        )

        DateTimeOutlinedField(
            modifier = Modifier.padding(horizontal = 16.dp),
            value = endDateTime,
            hint = stringResource(R.string.pick_finish_time_hint),
            onValueChange = { timestamp ->
                endTimeStamp = timestamp
            },
            isError = false
        )

        EnterStepsOutlinedField(
            modifier = Modifier.padding(horizontal = 16.dp),
            value = stepsValue,
            type = stepType,

```

```

        onTypeChange = { newType ->
            stepType = newType
        },
        onValueChange = { newStepsValue ->
            stepsValue = newStepsValue
        },
        isError = false
    )

    Button(
        modifier = Modifier
            .padding(horizontal = 16.dp)
            .fillMaxWidth()
            .height(height = 52.dp),
        onClick = {
            onSave.invoke(
                StepBatch(
                    steps = stepsValue.toInt(),
                    startTime = startTimeStamp!!,
                    endTime = endTimeStamp!!,
                    type = stepType
                )
            )
        },
        enabled = ctaEnabled
    ) {
        Text(
            text = stringResource(R.string.track),
            fontSize = 16.sp,
            fontWeight = FontWeight.Bold,
        )
    }
}
}
}
}

```

```
package com.zhiro.jetstep.feature.dashboard.presentation.dashboard.components
```

```

import androidx.compose.material3.DatePicker
import androidx.compose.material3.DatePickerDialog
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Text
import androidx.compose.material3.TextButton

```

```

import androidx.compose.material3.rememberDatePickerState
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.res.stringResource
import com.zhiro.jetstep.feature.dashboard.R
import java.util.Calendar
import java.util.Date

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DatePickerModal(
    onDateSelected: (Long) -> Unit,
    onDismiss: () -> Unit,
) {
    val datePickerState = rememberDatePickerState()
    var showTimePickerDialog by remember { mutableStateOf(false) }

    if (showTimePickerDialog) {
        TimePickerDialog(
            onDismiss = {
                showTimePickerDialog = false
            },
            onConfirm = { timePickerState ->
                val calendar = Calendar.getInstance()
                calendar.time =
                    Date(datePickerState.selectedDateMillis ?:
calendar.timeInMillis)
                calendar.set(Calendar.HOUR, timePickerState.hour)
                calendar.set(Calendar.MINUTE, timePickerState.minute)
                onDateSelected.invoke(calendar.time.time)
                showTimePickerDialog = false
            }
        )
    }
}

DatePickerDialog(
    onDismissRequest = onDismiss,
    confirmButton = {
        TextButton(

```

```

        onClick = {
            showTimePickerDialog = true
        }
    ) {
        Text(stringResource(R.string.ok))
    }
},
dismissButton = {
    TextButton(onClick = onDismiss) {
        Text(stringResource(R.string.cancel))
    }
}
) {
    DatePicker(state = datePickerState)
}
}

```

```
package com.zhiro.jetstep.feature.dashboard.presentation.dashboard.components
```

```

import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.defaultMinSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.LocalTextStyle
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.OutlinedTextFieldDefaults
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.text.input.KeyboardCapitalization
import androidx.compose.ui.text.input.KeyboardType

```

```

import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.zhiro.jetstep.feature.dashboard.R

const val TIME_PATTERN = "d MMM yyyy | HH:mm"

@Composable
fun DateTimeOutlinedField(
    modifier: Modifier = Modifier,
    value: String,
    hint: String,
    onValueChange: (Long) -> Unit,
    isError: Boolean,
) {
    var showEndDateTimePickerDialog by remember { mutableStateOf(false) }
    val defaultColors = OutlinedTextFieldDefaults.colors()

    if (showEndDateTimePickerDialog) {
        DatePickerModal(
            onDateSelected = { newTimestamp ->
                onValueChange.invoke(newTimestamp)
                showEndDateTimePickerDialog = false
            },
            onDismiss = {
                showEndDateTimePickerDialog = false
            }
        )
    }

    OutlinedTextField(
        modifier = modifier
            .fillMaxWidth()
            .defaultMinSize(minHeight = 52.dp)
            .clip(RoundedCornerShape(12.dp))
            .clickable {
                showEndDateTimePickerDialog = true
            },
        value = value,
        onValueChange = {},
        enabled = false,
        shape = RoundedCornerShape(12.dp),
        textStyle = LocalTextStyle.current.copy(

```

```

        fontSize = 16.sp,
        fontWeight = FontWeight.Medium,
        lineHeight = 19.sp
    ),
    maxLines = 1,
    singleLine = true,
    colors = OutlinedTextFieldDefaults.colors(
        disabledTextColor = defaultColors.unfocusedTextColor,
        disabledContainerColor = defaultColors.unfocusedContainerColor,
        disabledSupportingTextColor =
defaultColors.unfocusedSupportingTextColor,
        disabledTrailingIconColor =
defaultColors.unfocusedTrailingIconColor,
        disabledPlaceholderColor = defaultColors.unfocusedPlaceholderColor,
        disabledBorderColor = defaultColors.unfocusedTextColor
    ),
    keyboardOptions = KeyboardOptions(
        keyboardType = KeyboardType.Text,
        imeAction = ImeAction.Next,
        capitalization = KeyboardCapitalization.Words
    ),
    placeholder = {
        Text(
            text = hint,
            fontSize = 16.sp,
            lineHeight = 19.sp,
            fontWeight = FontWeight.Normal,
        )
    },
    isError = isError,
    trailingIcon = {
        IconButton(onClick = {
            showEndDateTimePickerDialog = true
        }) {
            Icon(
                modifier = Modifier.size(24.dp),
                painter = painterResource(R.drawable.ic_calendar_month),
                contentDescription = hint,
            )
        }
    }
}

```

```

    )
}

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard.components

import androidx.compose.foundation.layout.defaultMinSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.LocalTextStyle
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.text.isDigitsOnly
import com.zhiro.jetstep.core.ui.utils.icon
import com.zhiro.jetstep.domain.model.StepType
import com.zhiro.jetstep.domain.model.next
import com.zhiro.jetstep.feature.dashboard.R

@Composable
fun EnterStepsOutlinedField(
    modifier: Modifier = Modifier,
    value: String,
    type: StepType,
    onValueChange: (String) -> Unit,
    onTypeChange: (StepType) -> Unit,
    isError: Boolean,
) {

    OutlinedTextField(
        modifier = modifier
            .fillMaxWidth()

```

```

        .defaultMinSize(minHeight = 52.dp)
        .clip(RoundedCornerShape(12.dp)),
value = value,
onValueChange = { newValue ->
    if (newValue.isDigitsOnly() && newValue.length < 5) {
        onValueChange.invoke(newValue)
    }
},
shape = RoundedCornerShape(12.dp),
textStyle = LocalTextStyle.current.copy(
    fontSize = 16.sp,
    fontWeight = FontWeight.Medium,
    lineHeight = 19.sp
),
maxLines = 1,
singleLine = true,
keyboardOptions = KeyboardOptions(
    keyboardType = KeyboardType.Number,
    imeAction = ImeAction.Done
),
placeholder = {
    Text(
        text = stringResource(R.string.enter_steps_hint),
        fontSize = 16.sp,
        lineHeight = 19.sp,
        fontWeight = FontWeight.Normal,
    )
},
isError = isError,
trailingIcon = {
    IconButton(
        onClick = {
            onTypeChange.invoke(type.next())
        }
    ) {
        Icon(
            modifier = Modifier.size(24.dp),
            painter = type.icon,
            contentDescription =
stringResource(R.string.enter_steps_hint),
        )
    }
}

```

```

    }
)
}

```

```
package com.zhiro.jetstep.feature.dashboard.presentation.dashboard.components
```

```

import androidx.compose.animation.core.LinearOutSlowInEasing
import androidx.compose.animation.core.animateFloatAsState
import androidx.compose.animation.core.animateIntAsState
import androidx.compose.animation.core.tween
import androidx.compose.foundation.Canvas
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.offset
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableIntStateOf
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.rotate
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.StrokeCap
import androidx.compose.ui.graphics.drawscope.Stroke
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.Dp
import androidx.compose.ui.unit.IntOffset
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.zhiro.jetstep.core.ui.utils.color
import com.zhiro.jetstep.core.ui.utils.text

```

```
import com.zhiro.jetstep.domain.model.StepBatch
import com.zhiro.jetstep.domain.model.StepType
import com.zhiro.jetstep.domain.model.aggregate
```

```
@Composable
```

```
@Preview
```

```
internal fun PieChart(
    modifier: Modifier = Modifier,
    data: List<StepBatch> = listOf(
        StepBatch(
            steps = 500,
            startTime = 0, // Start from now
            endTime = 0, // 10 minutes later
            type = StepType.Walking
        ),
        StepBatch(
            steps = 750,
            startTime = 0, // Start from now
            endTime = 0, // 10 minutes later
            type = StepType.Running
        ),
        StepBatch(
            steps = 200,
            startTime = 0, // Start from now
            endTime = 0, // 10 minutes later
            type = StepType.Jogging
        ),
        StepBatch(
            steps = 1000,
            startTime = 0, // Start from now
            endTime = 0, // 10 minutes later
            type = StepType.Walking
        ),
        StepBatch(
            steps = 300,
            startTime = 0, // Start from now
            endTime = 0, // 10 minutes later
            type = StepType.Walking
        )
    ),
    radiusOuter: Dp = 140.dp,
    chartBarWidth: Dp = 35.dp,
```

```

    animDuration: Int = 1000,
) {
    val totalSum = data.sumOf(StepBatch::steps)
    val floatValue = mutableListOf<Float>()
    val aggregatedData by remember(data) {
        mutableStateOf(data.aggregate())
    }
    val stepsCount by remember(aggregatedData) {
        mutableIntStateOf(aggregatedData.values.sum())
    }
    // To set the value of each Arc according to
    // the value given in the data, we have used a simple formula.
    // For a detailed explanation check out the Medium Article.
    // The link is in the about section and readme file of this GitHub
Repository
    aggregatedData.values.forEachIndexed { index, steps ->
        floatValue.add(index, 360 * steps.toFloat() / totalSum.toFloat())
    }
    var animationPlayed by remember { mutableStateOf(false) }

    var lastValue = 0f

    // it is the diameter value of the Pie
    val animateSize by animateFloatAsState(
        targetValue = if (animationPlayed) radiusOuter.value * 2f else 0f,
        animationSpec = tween(
            durationMillis = animDuration,
            delayMillis = 0,
            easing = LinearOutSlowInEasing
        )
    )
    val animateStepsCount by animateIntAsState(
        targetValue = if (animationPlayed) stepsCount else 0,
        animationSpec = tween(
            durationMillis = animDuration, delayMillis = 0, easing =
LinearOutSlowInEasing
        )
    )

    val animateRotation by animateFloatAsState(
        targetValue = if (animationPlayed) 90f * 11f else 0f,
        animationSpec = tween(

```

```

        durationMillis = animDuration, delayMillis = 0, easing =
        LinearOutSlowInEasing
    )
)

LaunchedEffect(Unit) {
    animationPlayed = true
}

Column(
    modifier = modifier,
    horizontalAlignment = Alignment.CenterHorizontally
) {
    Box(
        modifier = Modifier.size(animateSize.dp),
        contentAlignment = Alignment.Center
    ) {
        Canvas(
            modifier = Modifier
                .offset { IntOffset.Zero }
                .size(radiusOuter * 2f)
                .rotate(animateRotation)
        ) {
            // draw each Arc for each data entry in Pie Chart
            floatValue.forEachIndexed { index, value ->
                drawArc(
                    color = aggregatedData.keys.elementAt(index).color,
                    startAngle = lastValue,
                    sweepAngle = value,
                    useCenter = false,
                    style = Stroke(chartBarWidth.toPx(), cap =
StrokeCap.Butt)
                )
                lastValue += value
            }
        }

        Text(
            modifier = Modifier.align(Alignment.Center),
            text = animateStepsCount.toString(),
            fontWeight = FontWeight.Medium,
            fontSize = 32.sp,

```

```

        color = Color.Black
    )
}

DetailsPieChart(
    aggregatedData = aggregatedData
)
}
}

@Composable
fun DetailsPieChart(aggregatedData: Map<StepType, Int>) {
    Column(
        modifier = Modifier
            .padding(top = 80.dp)
            .fillMaxWidth()
    ) {
        aggregatedData.values.forEachIndexed { index, value ->
            DetailsPieChartItem(
                data = Pair(aggregatedData.keys.elementAt(index).text, value),
                color = aggregatedData.keys.elementAt(index).color
            )
        }
    }
}

@Composable
fun DetailsPieChartItem(
    data: Pair<String, Int>,
    height: Dp = 45.dp,
    color: Color
) {
    Surface(
        modifier = Modifier
            .padding(vertical = 10.dp, horizontal = 40.dp),
        color = Color.Transparent
    ) {
        Row(
            modifier = Modifier.fillMaxWidth(),

```

```

        verticalAlignment = Alignment.CenterVertically
    ) {

        Box(
            modifier = Modifier
                .background(
                    color = color,
                    shape = RoundedCornerShape(10.dp)
                )
            .size(height)
        )

        Column(modifier = Modifier.fillMaxWidth()) {
            Text(
                modifier = Modifier.padding(start = 15.dp),
                text = data.first,
                fontWeight = FontWeight.Medium,
                fontSize = 22.sp,
                color = Color.Black
            )

            Text(
                modifier = Modifier.padding(start = 15.dp),
                text = data.second.toString(),
                fontWeight = FontWeight.Medium,
                fontSize = 22.sp,
                color = Color.Gray
            )
        }
    }

}

}

}

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard.components

import androidx.compose.material3.AlertDialog
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Text
import androidx.compose.material3.TextButton
import androidx.compose.material3.TimePicker
import androidx.compose.material3.TimePickerState

```

```

import androidx.compose.material3.rememberTimePickerState
import androidx.compose.runtime.Composable
import androidx.compose.ui.res.stringResource
import com.zhiro.jetstep.feature.dashboard.R
import java.util.Calendar

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun TimePickerDialog(
    onDismiss: () -> Unit,
    onConfirm: (TimePickerState) -> Unit,
) {

    val currentTime = Calendar.getInstance()

    val timePickerState = rememberTimePickerState(
        initialHour = currentTime.get(Calendar.HOUR_OF_DAY),
        initialMinute = currentTime.get(Calendar.MINUTE),
        is24Hour = true,
    )

    AlertDialog(
        onDismissRequest = onDismiss,
        dismissButton = {
            TextButton(onClick = onDismiss) {
                Text(stringResource(R.string.cancel))
            }
        },
        confirmButton = {
            TextButton(onClick = { onConfirm(timePickerState) }) {
                Text(stringResource(R.string.ok))
            }
        },
        text = {
            TimePicker(
                state = timePickerState,
            )
        }
    )
}

```

```

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard.usecase

```

```

import com.zhiro.jetstep.core.common.coroutines.AppDispatchers
import com.zhiro.jetstep.core.common.utils.atEndOfDay
import com.zhiro.jetstep.core.ui.base.mvi.SideEffectUseCase
import com.zhiro.jetstep.domain.repository.StepRepository
import
com.zhiro.jetstep.feature.dashboard.presentation.dashboard.DashboardAction
import com.zhiro.jetstep.feature.dashboard.presentation.dashboard.DashboardState
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.map
import java.time.LocalDate
import java.time.ZoneId

class LoadStepBatchesUseCase(
    dispatchers: AppDispatchers,
    stepRepository: StepRepository,
) : SideEffectUseCase<DashboardAction, DashboardState, DashboardAction.Load>(
    dispatchers = dispatchers
) {

    private val localDate = LocalDate.now()
    private val startOfDay =

localDate.atStartOfDay().atZone(ZoneId.systemDefault()).toInstant().toEpochMilli
()
    private val endOfDay =

localDate.atEndOfDay().atZone(ZoneId.systemDefault()).toInstant().toEpochMilli()

    override val sideEffectFlow: Flow<DashboardAction> =
stepRepository.loadStepBatches(
    startTime = startOfDay,
    endTime = endOfDay
).map(DashboardAction::Loaded)

    override suspend fun invoke(
        action: DashboardAction.Load,
        state: DashboardState,
    ): DashboardAction {
        return DashboardAction.None
    }

    override fun canHandle(action: DashboardAction): Boolean {

```

```

        return action is DashboardAction.Load && action.isLoading
    }
}

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard.usecase

import com.zhiro.jetstep.core.common.coroutines.AppDispatchers
import com.zhiro.jetstep.core.ui.base.mvi.UseCase
import com.zhiro.jetstep.domain.repository.StepRepository
import
com.zhiro.jetstep.feature.dashboard.presentation.dashboard.DashboardAction
import com.zhiro.jetstep.feature.dashboard.presentation.dashboard.DashboardState

class SaveStepBatchManuallyUseCase(
    dispatchers: AppDispatchers,
    private val stepRepository: StepRepository,
) : UseCase<DashboardAction, DashboardState, DashboardAction.SaveStepsManually>(
    dispatchers = dispatchers
) {
    override suspend fun invoke(
        action: DashboardAction.SaveStepsManually,
        state: DashboardState,
    ): DashboardAction {
        stepRepository.saveStepBatch(stepBatch = action.stepBatch)
        return DashboardAction.None
    }
}

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard

import com.zhiro.jetstep.core.ui.base.mvi.MviAction
import com.zhiro.jetstep.domain.model.StepBatch

sealed class DashboardAction : MviAction<DashboardState> {

    data object None : DashboardAction()

    data class Load(val isLoading: Boolean) : DashboardAction() {
        override fun accept(state: DashboardState): DashboardState {
            return state.copy(isLoading = isLoading)
        }
    }
}

```

```

data class SaveStepsManually(val stepBatch: StepBatch) : DashboardAction() {
    override fun accept(state: DashboardState): DashboardState {
        return state.copy(isLoading = true)
    }
}

data class Loaded(val stepBatches: List<StepBatch>) : DashboardAction() {
    override fun accept(state: DashboardState): DashboardState {
        return state.copy(
            steps = stepBatches,
            isLoading = false
        )
    }
}

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard

import androidx.navigation.NavGraphBuilder
import androidx.navigation.compose.composable
import com.zhiro.jetstep.core.navigation.routes.ScreenRoute

fun NavGraphBuilder.dashboardGraph() {
    composable<ScreenRoute.Dashboard> {
        DashboardScreen()
    }
}

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard

import com.zhiro.jetstep.core.ui.base.mvi.Reducer

internal class DashboardReducer : Reducer<DashboardAction, DashboardState> {

    override val initialState = DashboardState(
        isLoading = false,
        steps = listOf()
    )
}

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard

import androidx.compose.foundation.layout.Column

```

```

import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.systemBarsPadding
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.outlined.Edit
import androidx.compose.material3.FloatingActionButton
import androidx.compose.material3.Icon
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.lifecycle.compose.collectAsStateWithLifecycle
import
com.zhiro.jetstep.feature.dashboard.presentation.dashboard.components.AddStepsModal
import
com.zhiro.jetstep.feature.dashboard.presentation.dashboard.components.PieChart
import com.zhiro.jetstep.feature.locomotionclassifier.secondsValue
import org.koin.androidx.compose.koinViewModel

@Preview
@Composable
internal fun DashboardScreen(viewModel: DashboardViewModel = koinViewModel()) {
    val state by viewModel.state.collectAsStateWithLifecycle()
    var showAddNewStepsModal by remember { mutableStateOf(false) }

    Scaffold(
        floatingActionButton = {
            FloatingActionButton(
                onClick = {
                    showAddNewStepsModal = true

```

```

        },
    ) {
        Icon(Icons.Outlined.Edit, "add steps manually")
    }
}
) { padding ->

    if (showAddNewStepsModal) {

        AddStepsModal(
            onDismiss = {
                showAddNewStepsModal = false
            },
            onSave = { stepBatch ->
                viewModel.saveManually(stepBatch)
                showAddNewStepsModal = false
            }
        )
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(padding),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        PieChart(
            modifier = Modifier
                .systemBarsPadding()
                .padding(top = 32.dp)
                .fillMaxWidth(),
            data = state.steps
        )

        Text(
            text = secondsValue.value.toString(),
            fontWeight = FontWeight.Medium,
            fontSize = 32.sp,
            color = Color.Black
        )
    }
}

```

```

    }
}

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard

import com.zhiro.jetstep.domain.model.StepBatch

data class DashboardState(
    val isLoading: Boolean,
    val steps: List<StepBatch>
)

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard

import com.zhiro.jetstep.core.ui.base.mvi.StateMapper
import com.zhiro.jetstep.domain.model.StepBatch

internal data class DashboardUiState(
    val isLoading: Boolean,
    val steps: List<StepBatch>
) {

    companion object {

        fun stateMapper(): StateMapper<DashboardState, DashboardUiState> {
            return StateMapper { state ->
                DashboardUiState(
                    isLoading = state.isLoading,
                    steps = state.steps,
                )
            }
        }
    }
}

package com.zhiro.jetstep.feature.dashboard.presentation.dashboard

import com.zhiro.jetstep.core.common.coroutines.ModelCoroutineScope
import com.zhiro.jetstep.core.ui.base.mvi.BaseViewModel
import com.zhiro.jetstep.domain.model.StepBatch
import
com.zhiro.jetstep.feature.dashboard.presentation.dashboard.usecase.LoadStepBatch
esUseCase

```

```

import
com.zhiro.jetstep.feature.dashboard.presentation.dashboard.usecase.SaveStepBatch
ManuallyUseCase
import kotlinx.coroutines.ExperimentalCoroutinesApi
import kotlinx.coroutines.FlowPreview

@OptIn(FlowPreview::class, ExperimentalCoroutinesApi::class)
internal class DashboardViewModel(
    coroutineScope: ModelCoroutineScope,
    loadStepBatchesUseCase: LoadStepBatchesUseCase,
    saveStepBatchManuallyUseCase: SaveStepBatchManuallyUseCase
) : BaseViewModel<DashboardAction, DashboardState, DashboardUiState>(
    reducer = DashboardReducer(),
    stateMapper = DashboardUiState.stateMapper(),
    coroutineScope = coroutineScope,
    initialAction = DashboardAction.None,
    useCase = setOf(
        loadStepBatchesUseCase,
        saveStepBatchManuallyUseCase
    )
) {

    init {
        startLoading()
    }

    private fun startLoading() {
        action(DashboardAction.Load(isLoading = true))
    }

    fun saveManually(stepBatch: StepBatch) {
        action(DashboardAction.SaveStepsManually(stepBatch = stepBatch))
    }
}

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="960"
    android:viewportHeight="960"
    android:tint="?attr/colorControlNormal">
<path
    android:fillColor="@android:color/white"

```

```

        android:pathData="M200,880Q167,880 143.5,856.5Q120,833
120,800L120,240Q120,207 143.5,183.5Q167,160 200,160L240,160L240,120Q240,103
251.5,91.5Q263,80 280,80Q297,80 308.5,91.5Q320,103
320,120L320,160L640,160L640,120Q640,103 651.5,91.5Q663,80 680,80Q697,80
708.5,91.5Q720,103 720,120L720,160L760,160Q793,160 816.5,183.5Q840,207
840,240L840,800Q840,833 816.5,856.5Q793,880
760,880L200,880ZM200,800L760,800Q760,800 760,800Q760,800
760,800L760,400L200,400L200,800Q200,800 200,800Q200,800 200,800ZM480,560Q463,560
451.5,548.5Q440,537 440,520Q440,503 451.5,491.5Q463,480 480,480Q497,480
508.5,491.5Q520,503 520,520Q520,537 508.5,548.5Q497,560 480,560ZM320,560Q303,560
291.5,548.5Q280,537 280,520Q280,503 291.5,491.5Q303,480 320,480Q337,480
348.5,491.5Q360,503 360,520Q360,537 348.5,548.5Q337,560 320,560ZM640,560Q623,560
611.5,548.5Q600,537 600,520Q600,503 611.5,491.5Q623,480 640,480Q657,480
668.5,491.5Q680,503 680,520Q680,537 668.5,548.5Q657,560 640,560ZM480,720Q463,720
451.5,708.5Q440,697 440,680Q440,663 451.5,651.5Q463,640 480,640Q497,640
508.5,651.5Q520,663 520,680Q520,697 508.5,708.5Q497,720 480,720ZM320,720Q303,720
291.5,708.5Q280,697 280,680Q280,663 291.5,651.5Q303,640 320,640Q337,640
348.5,651.5Q360,663 360,680Q360,697 348.5,708.5Q337,720 320,720ZM640,720Q623,720
611.5,708.5Q600,697 600,680Q600,663 611.5,651.5Q623,640 640,640Q657,640
668.5,651.5Q680,663 680,680Q680,697 668.5,708.5Q657,720 640,720Z"/>
</vector>

```

```

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="960"
    android:viewportHeight="960"
    android:tint="?attr/colorControlNormal">
    <path
        android:fillColor="@android:color/white"
        android:pathData="M200,880Q167,880 143.5,856.5Q120,833
120,800L120,240Q120,207 143.5,183.5Q167,160 200,160L240,160L240,120Q240,103
251.5,91.5Q263,80 280,80Q297,80 308.5,91.5Q320,103
320,120L320,160L640,160L640,120Q640,103 651.5,91.5Q663,80 680,80Q697,80
708.5,91.5Q720,103 720,120L720,160L760,160Q793,160 816.5,183.5Q840,207
840,240L840,800Q840,833 816.5,856.5Q793,880
760,880L200,880ZM200,800L760,800Q760,800 760,800Q760,800
760,800L760,400L200,400L200,800Q200,800 200,800Q200,800 200,800ZM480,560Q463,560
451.5,548.5Q440,537 440,520Q440,503 451.5,491.5Q463,480 480,480Q497,480
508.5,491.5Q520,503 520,520Q520,537 508.5,548.5Q497,560 480,560ZM320,560Q303,560
291.5,548.5Q280,537 280,520Q280,503 291.5,491.5Q303,480 320,480Q337,480
348.5,491.5Q360,503 360,520Q360,537 348.5,548.5Q337,560 320,560ZM640,560Q623,560
611.5,548.5Q600,537 600,520Q600,503 611.5,491.5Q623,480 640,480Q657,480
668.5,491.5Q680,503 680,520Q680,537 668.5,548.5Q657,560 640,560ZM480,720Q463,720
451.5,708.5Q440,697 440,680Q440,663 451.5,651.5Q463,640 480,640Q497,640
508.5,651.5Q520,663 520,680Q520,697 508.5,708.5Q497,720 480,720ZM320,720Q303,720
291.5,708.5Q280,697 280,680Q280,663 291.5,651.5Q303,640 320,640Q337,640
348.5,651.5Q360,663 360,680Q360,697 348.5,708.5Q337,720 320,720ZM640,720Q623,720
611.5,708.5Q600,697 600,680Q600,663 611.5,651.5Q623,640 640,640Q657,640
668.5,651.5Q680,663 680,680Q680,697 668.5,708.5Q657,720 640,720Z"/>

```

```

668.5,491.5Q680,503 680,520Q680,537 668.5,548.5Q657,560 640,560ZM480,720Q463,720
451.5,708.5Q440,697 440,680Q440,663 451.5,651.5Q463,640 480,640Q497,640
508.5,651.5Q520,663 520,680Q520,697 508.5,708.5Q497,720 480,720ZM320,720Q303,720
291.5,708.5Q280,697 280,680Q280,663 291.5,651.5Q303,640 320,640Q337,640
348.5,651.5Q360,663 360,680Q360,697 348.5,708.5Q337,720 320,720ZM640,720Q623,720
611.5,708.5Q600,697 600,680Q600,663 611.5,651.5Q623,640 640,640Q657,640
668.5,651.5Q680,663 680,680Q680,697 668.5,708.5Q657,720 640,720Z"/>

```

```
</vector>
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
```

```
    <string name="ok">Ок</string>
```

```
    <string name="cancel">Скасувати</string>
```

```
    <string name="pick_start_time_hint">Виберіть час початку</string>
```

```
    <string name="pick_finish_time_hint">Виберіть час фінішу</string>
```

```
    <string name="enter_steps_hint">Введіть кроки</string>
```

```
    <string name="track">Зберіть</string>
```

```
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest></manifest>
```

```
plugins {
```

```
    alias(libs.plugins.jetstep.android.feature)
```

```
    alias(libs.plugins.jetstep.koin)
```

```
}
```

```
android {
```

```
    namespace = "com.zhiro.jetstep.feature.dashboard"
```

```
}
```

```
dependencies {
```

```
    implementation(projects.core.common)
```

```
    implementation(libs.onnxruntime.android)
```

```
}
```

```
package com.zhiro.jetstep.feature.locomotionclassifier.di.module
```

```
import com.zhiro.jetstep.feature.locomotionclassifier.LocomotionClassifier
```

```
import com.zhiro.jetstep.feature.locomotionclassifier.LocomotionClassifierImpl
```

```
import com.zhiro.jetstep.feature.locomotionclassifier.LocomotionHelper
```

```
import com.zhiro.jetstep.feature.locomotionclassifier.LocomotionHelperImpl
```

```
import org.koin.core.definition.BeanDefinition
```

```
import org.koin.core.module.dsl.bind
```

```

import org.koin.core.module.dsl.factoryOf
import org.koin.core.module.dsl.singleOf
import org.koin.dsl.module

val featureLocomotionHelperModule
    get() = module {
        singleOf(::LocomotionHelperImpl, BeanDefinition<LocomotionHelper>::bind)
        factoryOf(::LocomotionClassifierImpl,
        BeanDefinition<LocomotionClassifier>::bind)
    }

package com.zhiro.jetstep.feature.locomotionclassifier.model.step1.record

internal data class TimestampAccelerometerRecord(
    val x: Float,
    val y: Float,
    val z: Float,
    val collectedAtMillis: Long,
) {
    companion object {

        fun create(x: Float, y: Float, z: Float) = TimestampAccelerometerRecord(
            x = x,
            y = y,
            z = z,
            collectedAtMillis = System.currentTimeMillis()
        )

        fun create(x: Float, y: Float, z: Float, collectedAtMillis: Long = 0,
        pace: String) = TimestampAccelerometerRecord(
            x = x,
            y = y,
            z = z,
            collectedAtMillis = collectedAtMillis
        )
    }
}

package com.zhiro.jetstep.feature.locomotionclassifier.model.step1.record

internal data class TimestampStepRecord(
    val aggregatedStepsCount: Long,
    val collectedAtMillis: Long
) {

```

```

companion object {

    fun create(aggregatedStepsCount: Long) = TimestampStepRecord(
        aggregatedStepsCount = aggregatedStepsCount,
        collectedAtMillis = System.currentTimeMillis()
    )
}

}

package com.zhiro.jetstep.feature.locomotionclassifier.model.step2

data class AccelerometerMagnitude(
    val collectedAtMillis: Long,
    val magnitude: Float
)

package com.zhiro.jetstep.feature.locomotionclassifier.model.step2

import
com.zhiro.jetstep.feature.locomotionclassifier.model.step1.record.TimestampAccelerometerRecord
import
com.zhiro.jetstep.feature.locomotionclassifier.model.step1.record.TimestampStepRecord
import
com.zhiro.jetstep.feature.locomotionclassifier.model.step4.ValleyAndPeakRelation
import com.zhiro.jetstep.feature.locomotionclassifier.model.step5.StepFeature
import
com.zhiro.jetstep.feature.locomotionclassifier.utils.algorithms.findLocalPeaksAndValleysFilter
import
com.zhiro.jetstep.feature.locomotionclassifier.utils.algorithms.findPeakSimple
import
com.zhiro.jetstep.feature.locomotionclassifier.utils.algorithms.findRelatedPeaksAndValleysFilter
import
com.zhiro.jetstep.feature.locomotionclassifier.utils.algorithms.mapLowPassFilter
import
com.zhiro.jetstep.feature.locomotionclassifier.utils.algorithms.toFeatures
import com.zhiro.jetstep.feature.locomotionclassifier.utils.printDuration
import com.zhiro.jetstep.feature.locomotionclassifier.utils.sumOf
import kotlin.math.sqrt

```

```

internal data class StepsDetails(
    val stepsAmount: Long,
    val accelerometerRecords: List<TimestampAccelerometerRecord>,
) {

    // Step 1
    private val accelerometerMagnitudeRecords: List<AccelerometerMagnitude>
get() {
    return printDuration {
        val rmags = accelerometerRecords.map { accelerometerRecord ->
            AccelerometerMagnitude(
                collectedAtMillis =
accelerometerRecord.collectedAtMillis,
                magnitude = sqrt(
                    accelerometerRecord.x * accelerometerRecord.x +
                    accelerometerRecord.y * accelerometerRecord.y +
                    accelerometerRecord.z * accelerometerRecord.z
                )
            )
        }

        val gravity = rmags.sumOf(AccelerometerMagnitude::magnitude) /
rmags.size

        rmags.map { rmag ->
            rmag.copy(magnitude = rmag.magnitude - gravity)
        }.mapLowPassFilter(alpha = LOW_PASS_FILTER_ALPHA)
    }
}

    // Step 2
    private val accelerometerMagnitudePeaksAndValleysRecords:
List<AccelerometerMagnitude> get() {
    return printDuration {
        val accelerometerMagnitudeRecords =
accelerometerMagnitudeRecords
        val peaks =
accelerometerMagnitudeRecords.findPeakSimple(threshold = UPPER_THRESHOLD)
        val valleys =
accelerometerMagnitudeRecords.findPeakSimple(threshold = LOWER_THRESHOLD,
reversed = true)
        (peaks +

```

```

valleys).sortedBy(AccelerometerMagnitude::collectedAtMillis).also {
    println("accelerometerMagnitudePeaksAndValleysRecords size:
    ${it.size}")
}
}

// Step 3
private val accelerometerMagnitudeLocalPeaksAndValleysRecords:
List<AccelerometerMagnitude> get() {
    return printDuration {

        accelerometerMagnitudePeaksAndValleysRecords.findLocalPeaksAndValleysFilter(
            intervalDurationWindow = LOCAL_INTERVAL_DURATION_WINDOW
        ).also {
            println("accelerometerMagnitudeLocalPeaksAndValleysRecords
size: ${it.size}")
        }
    }
}

// Step 4
private val valleyPeakRelationRecords: List<ValleyAndPeakRelation> get() {
    return printDuration {

        accelerometerMagnitudeLocalPeaksAndValleysRecords.findRelatedPeaksAndValleysFilter(
            intervalDurationWindow = STEP_WINDOW_DURATION,
            valleyPeakDurationWindow = VALLEY_PEAK_DURATION_WINDOW,
            lowerThreshold = LOWER_THRESHOLD,
            upperThreshold = UPPER_THRESHOLD
        ).also {
            println("valleyPeakRelationRecords size: ${it.size}")
        }
    }
}

// Step 5
val stepFeatures: List<StepFeature> get() {
    return printDuration {
        valleyPeakRelationRecords.toFeatures(error =
STEP_INTERVAL_ERROR).also {

```

```

        println("stepFeatures size: ${it.size}")
    }
}

companion object {

    private const val STEP_WINDOW_DURATION = 500L
    private const val VALLEY_PEAK_DURATION_WINDOW = 500L
    private const val LOCAL_INTERVAL_DURATION_WINDOW = 300L
    private const val STEP_INTERVAL_ERROR = 100L
    private const val UPPER_THRESHOLD = 5f
    private const val LOWER_THRESHOLD = -2f
    private const val LOW_PASS_FILTER_ALPHA = 0.9f

    fun create(
        timestampStepRecords: List<TimestampStepRecord>,
        timestampAccelerometerRecords: List<TimestampAccelerometerRecord>,
    ) = printDuration {
        StepsDetails(
            stepsAmount = run {
                val last = timestampStepRecords.lastOrNull()
                val first = timestampStepRecords.firstOrNull()

                last?.aggregatedStepsCount?.minus(first?.aggregatedStepsCount ?: 0L) ?: 0L
            },
            accelerometerRecords = timestampAccelerometerRecords
        )
    }
}

package com.zhiro.jetstep.feature.locomotionclassifier.model.step4

import
com.zhiro.jetstep.feature.locomotionclassifier.model.step2.AccelerometerMagnitude
import kotlin.math.abs

data class ValleyAndPeakRelation(
    val peak: AccelerometerMagnitude,

```

```

    val valley: AccelerometerMagnitude,
    val duration: Long
) {
    companion object {

        fun create(
            peak: AccelerometerMagnitude,
            valley: AccelerometerMagnitude,
        ) = ValleyAndPeakRelation(
            peak = peak,
            valley = valley,
            duration = abs(peak.collectedAtMillis - valley.collectedAtMillis)
        )
    }
}

package com.zhiro.jetstep.feature.locomotionclassifier.model.step5

import
com.zhiro.jetstep.feature.locomotionclassifier.model.step4.ValleyAndPeakRelation

data class StepFeature(
    val magnitude: Float,
    val peakValleyIntervalDuration: Long,
    val currentPreviousPeakIntervalDuration: Long,
    val timestamp: Long
) {

    companion object {

        fun create(
            prevValleyAndPeakRelation: ValleyAndPeakRelation,
            currentValleyAndPeakRelation: ValleyAndPeakRelation,
        ) = StepFeature(
            magnitude = currentValleyAndPeakRelation.peak.magnitude,
            peakValleyIntervalDuration = currentValleyAndPeakRelation.duration,
            currentPreviousPeakIntervalDuration =
currentValleyAndPeakRelation.peak.collectedAtMillis -
                prevValleyAndPeakRelation.peak.collectedAtMillis,
            timestamp = currentValleyAndPeakRelation.peak.collectedAtMillis
        )
    }
}

```

```

}

fun List<StepFeature>.toFloatArray() = map { stepFeature ->
    listOf(
        stepFeature.currentPreviousPeakIntervalDuration.toFloat(),
        stepFeature.peakValleyIntervalDuration.toFloat(),
        stepFeature.magnitude
    )
}.flatten().toFloatArray()

package com.zhiro.jetstep.feature.locomotionclassifier.model

import com.zhiro.jetstep.feature.locomotionclassifier.model.step2.StepsDetails

data class ClassifiedSteps(
    val steps: Long,
    val typeId: Long,
    val startTime: Long,
    val endTime: Long,
) {

    companion object {

        internal fun create(stepsDetails: StepsDetails, typeId: Long) =
ClassifiedSteps(
            typeId = typeId,
            steps = stepsDetails.stepsAmount,
            startTime =
stepsDetails.accelerometerRecords.first().collectedAtMillis,
            endTime = stepsDetails.accelerometerRecords.last().collectedAtMillis
        )
    }
}

package com.zhiro.jetstep.feature.locomotionclassifier.utils.algorithms

import
com.zhiro.jetstep.feature.locomotionclassifier.model.step2.AccelerometerMagnitud
e

fun List<AccelerometerMagnitude>.findLocalPeaksAndValleysFilter(
    intervalDurationWindow: Long
): List<AccelerometerMagnitude> {

```

```

if (isEmpty()) return emptyList()
val startTimestamp = first().collectedAtMillis
val endTimestamp = last().collectedAtMillis
val localPeaksAndValleys = mutableListOf<AccelerometerMagnitude>()

var localEndTimestamp = startTimestamp
while (startTimestamp <= endTimestamp) {
    val localStartTimestamp = localEndTimestamp + 1
    localEndTimestamp = localStartTimestamp + intervalDurationWindow

    val local = filter { mag ->
        mag.collectedAtMillis in localStartTimestamp..localEndTimestamp
    }

    if (local.isEmpty()) {
        val nextLocalStartTimestamp = firstOrNull { mag ->
            localEndTimestamp < mag.collectedAtMillis
        }?.collectedAtMillis
        if (nextLocalStartTimestamp == null) {
            break
        } else {
            localEndTimestamp = nextLocalStartTimestamp
        }
    } else {
        val localMin = local.minBy(AccelerometerMagnitude::magnitude)
        val localMax = local.maxBy(AccelerometerMagnitude::magnitude)

        if (localMin.collectedAtMillis < localMax.collectedAtMillis) {
            localPeaksAndValleys.add(localMin)
            localPeaksAndValleys.add(localMax)
        } else {
            localPeaksAndValleys.add(localMax)
            localPeaksAndValleys.add(localMin)
        }
    }
}
return localPeaksAndValleys
}

package com.zhiro.jetstep.feature.locomotionclassifier.utils.algorithms

import
com.zhiro.jetstep.feature.locomotionclassifier.model.step2.AccelerometerMagnitud

```

```

e
import java.util.stream.IntStream.range

fun List<AccelerometerMagnitude>.findPeakSimple(
    threshold: Float,
    reversed: Boolean = false,
): List<AccelerometerMagnitude> {
    val filteredValues = mutableListOf<AccelerometerMagnitude>()
    val coef = if (reversed) -1 else 1

    @Suppress("NAME_SHADOWING")
    val threshold = threshold * coef

    for (index in range(0, size)) {
        val prev = (getOrNull(index - 1)?.magnitude ?: Float.MIN_VALUE) * coef
        val current = get(index).magnitude * coef
        val next = (getOrNull(index + 1)?.magnitude ?: Float.MIN_VALUE) * coef

        when {
            prev > current && current < next && current <= threshold -> {
                filteredValues.add(get(index))
            }

            else -> continue
        }
    }

    return filteredValues
}

package com.zhiro.jetstep.feature.locomotionclassifier.utils.algorithms

import
com.zhiro.jetstep.feature.locomotionclassifier.model.step2.AccelerometerMagnitud
e
import
com.zhiro.jetstep.feature.locomotionclassifier.model.step4.ValleyAndPeakRelation

fun List<AccelerometerMagnitude>.findRelatedPeaksAndValleysFilter(
    intervalDurationWindow: Long,
    valleyPeakDurationWindow: Long,
    lowerThreshold: Float,

```

```

    upperThreshold: Float,
): List<ValleyAndPeakRelation> {
    if (isEmpty()) return emptyList()
    val startTimestamp = first().collectedAtMillis
    val endTimestamp = last().collectedAtMillis
    val valleyAndPeakRelations = mutableListOf<ValleyAndPeakRelation>()

    var localEndTimestamp = startTimestamp
    while (startTimestamp <= endTimestamp) {
        val localStartTimestamp = localEndTimestamp + 1
        localEndTimestamp = localStartTimestamp + intervalDurationWindow

        val localValleys = filter { mag ->
            mag.collectedAtMillis in localStartTimestamp..localEndTimestamp &&
mag.magnitude <= lowerThreshold
        }
        val localMin =
localValleys.minByOrNull(AccelerometerMagnitude::magnitude)

        if (localMin == null) {
            val nextLocalStartTimestamp = firstOrNull { mag ->
                localEndTimestamp < mag.collectedAtMillis
            }?.collectedAtMillis
            if (nextLocalStartTimestamp == null) {
                break
            } else {
                localEndTimestamp = nextLocalStartTimestamp
                continue
            }
        }

        val localPeaks = filter { mag ->
            mag.collectedAtMillis in (
                localMin.collectedAtMillis + 1..localMin.collectedAtMillis +
valleyPeakDurationWindow
            ) && mag.magnitude >= upperThreshold
        }
        val localMax = localPeaks.maxByOrNull(AccelerometerMagnitude::magnitude)

        if (localMax != null) {
            val valleyAndPeakRelation = ValleyAndPeakRelation.create(
                peak = localMax,

```

```

        valley = localMin
    )
    valleyAndPeakRelations.add(valleyAndPeakRelation)
}
}
return valleyAndPeakRelations
}

package com.zhironet.jetstep.feature.locomotionclassifier.utils.algorithms

import
com.zhironet.jetstep.feature.locomotionclassifier.model.step2.AccelerometerMagnitude

fun List<AccelerometerMagnitude>.mapLowPassFilter(alpha: Float):
List<AccelerometerMagnitude> {
    return mapIndexed { index, accelerometerMagnitude ->
        if (index == 0) {
            accelerometerMagnitude
        } else {
            accelerometerMagnitude.copy(
                magnitude = (alpha * get(index - 1).magnitude) + (1 - alpha) *
accelerometerMagnitude.magnitude
            )
        }
    }
}

package com.zhironet.jetstep.feature.locomotionclassifier.utils.algorithms

fun <T, K : Number> List<T>.mode(keySelector: (T) -> K) =
    groupBy(keySelector).eachCount().maxByOrNull(Map.Entry<K,
Int>::value)?.key

fun <T, K : Number> List<T>.valueCounts(keySelector: (T) -> K) =
    groupBy(keySelector).eachCount()

package com.zhironet.jetstep.feature.locomotionclassifier.utils.algorithms

import
com.zhironet.jetstep.feature.locomotionclassifier.model.step4.ValleyAndPeakRelation
import com.zhironet.jetstep.feature.locomotionclassifier.model.step5.StepFeature
import com.zhironet.jetstep.feature.locomotionclassifier.utils.round

```

```

import java.util.stream.IntStream.range

fun List<ValleyAndPeakRelation>.toFeatures(error: Long): List<StepFeature> {
    val stepFeatures = mutableListOf<StepFeature>()
    // creating features
    for (index in range(0, size)) {
        val prev = getOrNull(index - 1)
        val current = get(index)
        if (prev != null && current.peak.collectedAtMillis -
prev.peak.collectedAtMillis > 0) {
            stepFeatures.add(
                StepFeature.create(
                    prevValleyAndPeakRelation = prev,
                    currentValleyAndPeakRelation = current
                )
            )
        }
    }

    // finding step interval duration mode
    val peaksIntervalDurationMode = stepFeatures.mode { stepFeature ->
        stepFeature.currentPreviousPeakIntervalDuration.round(2)
    } ?: Long.MIN_VALUE

    // filtering features
    val filteredStepFeatures = stepFeatures.filter { stepFeature ->
        stepFeature.currentPreviousPeakIntervalDuration >=
peaksIntervalDurationMode - error &&
        stepFeature.currentPreviousPeakIntervalDuration <=
peaksIntervalDurationMode + error
    }

    return filteredStepFeatures
}

package com.zhiro.jetstep.feature.locomotionclassifier.utils

internal inline fun <T> Iterable<T>.sumOf(selector: (T) -> Float): Float {
    var sum = 0f
    for (element in this) {
        sum += selector(element)
    }
}

```

```

        return sum
    }

package com.zhiro.jetstep.feature.locomotionclassifier.utils

import kotlin.math.abs

fun Long.round(decimals: Int): Long {
    var multiplier = 1.0
    repeat(abs(decimals)) { multiplier *= 10 }

    return (kotlin.math.round(this / multiplier) * multiplier).toLong()
}

package com.zhiro.jetstep.feature.locomotionclassifier.utils

internal inline fun <R> printDuration(tag: String = "TAG!!!", block: () -> R): R
{
    val startTime = System.currentTimeMillis()
    val blockResult = block.invoke()
    val endTime = System.currentTimeMillis()
    println("$tag duration: ${endTime - startTime}")
    return blockResult
}

package com.zhiro.jetstep.feature.locomotionclassifier

import ai.onnxruntime.OnnxTensor
import ai.onnxruntime.OrtEnvironment
import ai.onnxruntime.OrtSession
import android.content.Context
import com.zhiro.jetstep.feature.dashboard.R
import com.zhiro.jetstep.feature.locomotionclassifier.model.ClassifiedSteps
import com.zhiro.jetstep.feature.locomotionclassifier.model.step2.StepsDetails
import com.zhiro.jetstep.feature.locomotionclassifier.model.step5.toFloatArray
import java.nio.FloatBuffer

internal interface LocomotionClassifier {

    fun predict(stepsDetails: StepsDetails): ClassifiedSteps?
}

internal class LocomotionClassifierImpl(private val applicationContext: Context,
) : LocomotionClassifier {

```

```

override fun predict(stepsDetails: StepsDetails): ClassifiedSteps? {
    if (stepsDetails.stepsAmount < 5) return null
    val inputs = stepsDetails.stepFeatures.toFloatArray()
    if (inputs.isEmpty()) return null

    val ortEnvironment = OrtEnvironment.getEnvironment()
    val ortSessionScaler = createORTSessionScaler(ortEnvironment)
    val outputScaler = runScaler(inputs, ortSessionScaler, ortEnvironment)

    val inputClassifier =
outputScaler.toList().map(FloatArray::toList).flatten().toFloatArray()
    val ortSessionClassifier = createORTSessionClassifier(ortEnvironment)
    val output = runPrediction(inputClassifier, ortSessionClassifier,
ortEnvironment)
        .toList()
        .groupBy { it }
        .eachCount()
        .maxByOrNull(Map.Entry<Long, Int>::value)
        ?.key

    return if (output != null) {
        ClassifiedSteps.create(
            stepsDetails = stepsDetails,
            typeId = output
        )
    } else {
        null
    }
}

// Make predictions with given inputs
private fun runPrediction(
    input: FloatArray,
    ortSession: OrtSession,
    ortEnvironment: OrtEnvironment,
): LongArray {
    // Get the name of the input node
    val inputName = ortSession.inputNames?.iterator()?.next()
    // Make a FloatBuffer of the inputs
    val floatBufferInputs = FloatBuffer.wrap(input)
    val featureArraysCount = input.size / FEATURES_COUNT

```

```

        val inputTensor = OnnxTensor.createTensor(
            ortEnvironment,
            floatBufferInputs,
            longArrayOf(featureArraysCount, FEATURES_COUNT)
        )
        // Run the model
        val results = ortSession.run(mapOf(inputName to inputTensor))
        // Fetch and return the results
        return results[0].value as LongArray
    }

    // Make predictions with given inputs
    private fun runScaler(
        input: FloatArray,
        ortSession: OrtSession,
        ortEnvironment: OrtEnvironment,
    ): Array<FloatArray> {
        // Get the name of the input node
        val inputName = ortSession.inputNames?.iterator()?.next()
        // Make a FloatBuffer of the inputs
        val floatBufferInputs = FloatBuffer.wrap(input)
        val featureArraysCount = input.size / FEATURES_COUNT

        val inputTensor = OnnxTensor.createTensor(
            ortEnvironment,
            floatBufferInputs,
            longArrayOf(featureArraysCount, FEATURES_COUNT)
        )
        // Run the model
        val results = ortSession.run(mapOf(inputName to inputTensor))
        // Fetch and return the results
        return results[0].value as Array<FloatArray>
    }

    // Create an OrtSession
    private fun createORTSessionClassifier(ortEnvironment: OrtEnvironment):
OrtSession {
        val modelBytes =

applicationContext.resources.openRawResource(R.raw.sklearn_model).readBytes()
        return ortEnvironment.createSession(modelBytes)
    }

```

```

    }

    private fun createORTSessionScaler(ortEnvironment: OrtEnvironment):
OrtSession {
        val modelBytes =

applicationContext.resources.openRawResource(R.raw.sklearn_scaler).readBytes()
        return ortEnvironment.createSession(modelBytes)
    }

    companion object {
        const val FEATURES_COUNT = 3L
    }
}

package com.zhiro.jetstep.feature.locomotionclassifier

import androidx.compose.runtime.mutableStateOf
import com.zhiro.jetstep.core.common.coroutines.ModelCoroutineScope
import com.zhiro.jetstep.feature.locomotionclassifier.model.ClassifiedSteps
import
com.zhiro.jetstep.feature.locomotionclassifier.model.step1.record.TimestampAccel
erometerRecord
import
com.zhiro.jetstep.feature.locomotionclassifier.model.step1.record.TimestampStepR
ecord
import com.zhiro.jetstep.feature.locomotionclassifier.model.step2.StepsDetails
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.SharedFlow
import kotlinx.coroutines.launch
import kotlinx.coroutines.sync.Mutex
import kotlinx.coroutines.sync.withLock
import kotlin.time.Duration.Companion.seconds

var secondsValue = mutableStateOf(
    0L
)

interface LocomotionHelper {

    val lastClassifiedStepsFlow: SharedFlow<ClassifiedSteps>

    fun feedSteps(aggregatedStepsCount: Long)

```

```

    fun feedAccelerometer(x: Float, y: Float, z: Float)
}

internal class LocomotionHelperImpl(
    private val coroutineScope: ModelCoroutineScope,
    private val locomotionClassifier: LocomotionClassifier
) : LocomotionHelper {

    override val lastClassifiedStepsFlow = MutableSharedFlow<ClassifiedSteps>()
    private val timestampStepRecords = mutableListOf<TimestampStepRecord>()
    private val timestampAccelerometerRecords =
mutableListOf<TimestampAccelerometerRecord>()
    private var lastTimeAggregatedAtMillis = System.currentTimeMillis()
    private val mutex = Mutex()

    override fun feedSteps(aggregatedStepsCount: Long) {
        // assign time to a record and save
        coroutineScope.launch {
            val timestampStep =
                TimestampStepRecord.create(aggregatedStepsCount =
aggregatedStepsCount)
            mutex.withLock(this) {
                timestampStepRecords.add(timestampStep)
            }
            startAggregationIfRequired()
        }
    }

    override fun feedAccelerometer(x: Float, y: Float, z: Float) {
        // assign time to a record and save
        coroutineScope.launch {
            val timestampAccelerometerRecord =
                TimestampAccelerometerRecord.create(x = x, y = y, z = z)
            mutex.withLock(this) {
                timestampAccelerometerRecords.add(timestampAccelerometerRecord)
            }
            startAggregationIfRequired()
        }
    }

    /**

```

```

    * starts aggregation every [AGGREGATION_COOLDOWN]
    */
    private suspend fun startAggregationIfRequired() {
        secondsValue.value = (System.currentTimeMillis() -
lastTimeAggregatedAtMillis) / 1000
        if (System.currentTimeMillis() - lastTimeAggregatedAtMillis >
AGGREGATION_COOLDOWN.inWholeMilliseconds) {

            // start aggregation
            var tempTimestampStepRecords: List<TimestampStepRecord>
            var tempTimestampAccelerometerRecords:
List<TimestampAccelerometerRecord>
            mutex.withLock {
                tempTimestampAccelerometerRecords =
timestampAccelerometerRecords.toList()
                tempTimestampStepRecords = timestampStepRecords.toList()
                timestampAccelerometerRecords.clear()
                timestampStepRecords.clear()
                // update lastTimeAggregatedAtMillis and release the lock
                lastTimeAggregatedAtMillis = System.currentTimeMillis()
            }
            val stepsDetails = StepsDetails.create(
                timestampStepRecords = tempTimestampStepRecords,
                timestampAccelerometerRecords =
tempTimestampAccelerometerRecords
            )
            val classifiedStep = locomotionClassifier.predict(stepsDetails =
stepsDetails)
            if (classifiedStep != null) {
                lastClassifiedStepsFlow.emit(classifiedStep)
            }
        }
    }

    companion object {
        private val AGGREGATION_COOLDOWN = 60.seconds
    }
}

```

ДОДАТОК Б
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

«Дослідження методів та розроблення засобів визначення руху для
мобільних застосунків.
Research of Methods and Development of Motion Detection Tools for Mobile Applications»

Виконав:
ст. гр. КНТ-219

Керівник:
професор, к.т.н.

Єгор ЖИР

Галина ТАБУНЩИК

Рисунок Б.1 – Слайд 1

Актуальність, мета роботи, об'єкт та предмет дослідження

JetStep

● **Мета роботи** полягає у відстеженні фізичної активності користувача в застосунку.

■ **Об'єкт дослідження** – засоби визначення руху.

★ **Предмет дослідження** – дослідження методів та розроблення засобів визначення руху для мобільних застосунків.

2

Рисунок Б.2 – Слайд 2

Завдання дипломної роботи

- **Перше завдання**
Дослідити механіку руху.
- **Друге завдання**
Вивчити методи розпізнавання руху.
- **Третє завдання**
Вивчити методи розпізнавання руху.
- **Четверте завдання**
Оформити пояснювальну записку.

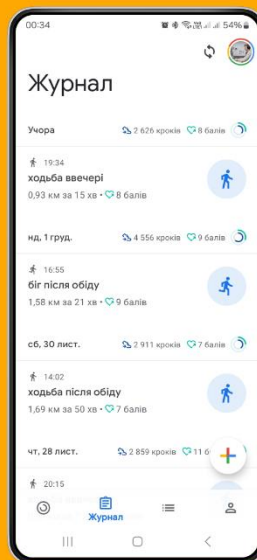
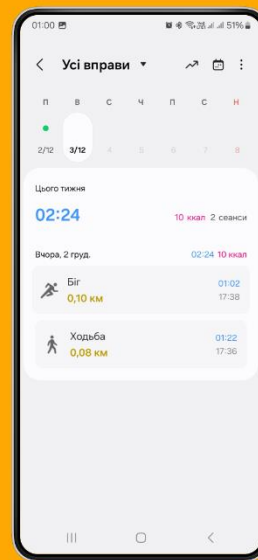
3

Рисунок Б.3 – Слайд 3

Аналіз існуючих додатків

Google Fit та Samsung – це застасунки для відстеження здоров'я та фізичної активності. Вони дозволяють користувачам моніторити свої показники фітнес даних, такі як кроки, калорії, фізична активність та інші (рис. 1, 2).

JetStep

Рисунок 1 –
Застосунок Google FitРисунок 2 –
Застосунок Samsung Health

4

Рисунок Б.4 – Слайд 4

Функціонал розробленого застосунку

- Відображення кількості кроків за день.
- Класифікація та відображення видів руху.
- Збирання даних під час неактивності застосунку.
- Збереження даних про кількість кроків.
- Можливість додавання кроків вручну.
- Обфускація APK.
- Заборона доступу до внутрішньої пам'яті застосунку.

5

Рисунок Б.5 – Слайд 5

Аналіз предметної області

ХОДЬБА



Рисунок 3 – Механіка ходьби

- **Присутня циклічність.**
- Невелика довжина кроку.
- Невелика висота кроку.
- Торкання стопи з перекатом.
- Помірна тривалість торкання.
- **Помірна частота кроків.**
- Перпендикулярне положення корпусу.
- Невелика амплітуда руки.

БІГ

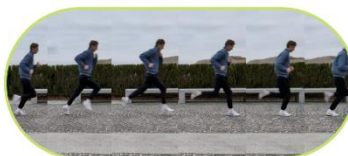


Рисунок 4 – Механіка бігу

- **Присутня циклічність.**
- Велика довжина кроку.
- Помірна висота кроку.
- Торкання передньою частиною стопи без перекаату.
- Нетривале торкання.
- **Невисока частота кроків.**
- Під нахилом положення корпусу.
- Велика амплітуда руки.

БІГ ПІДТЮПЦЕМ

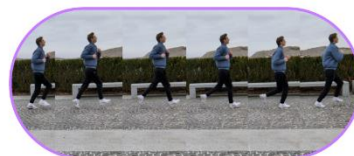
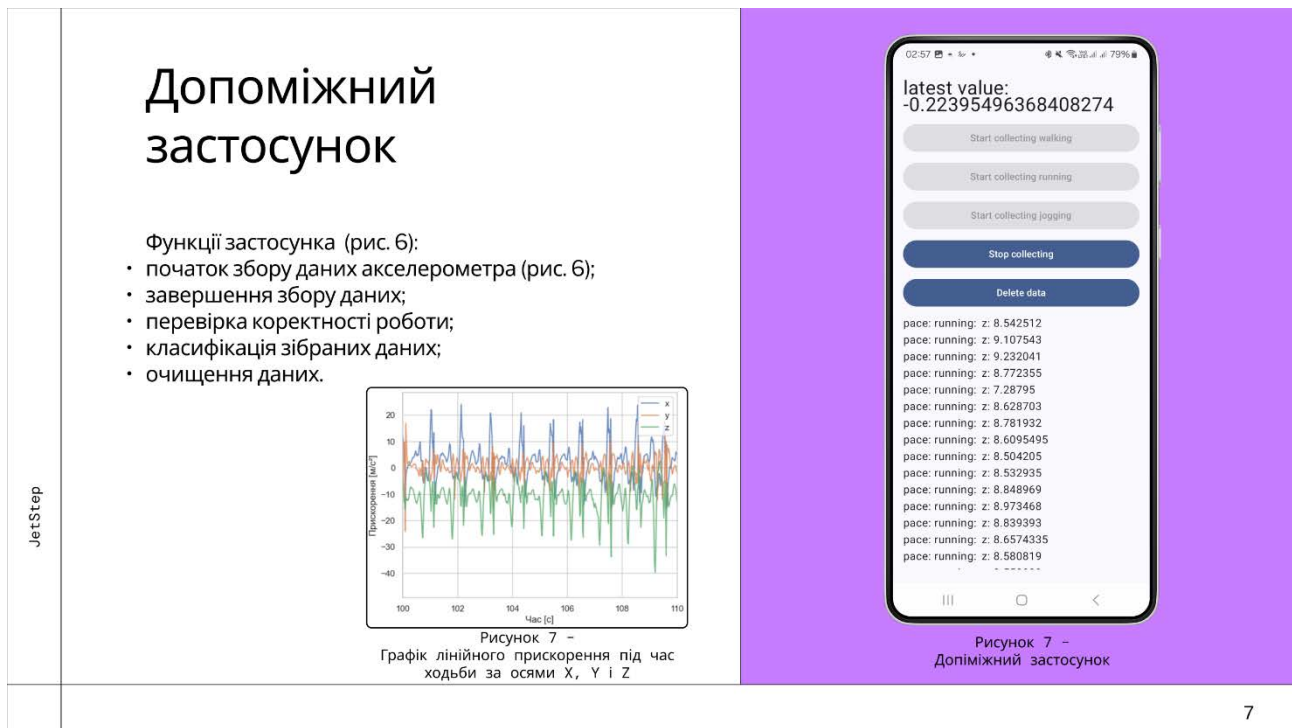


Рисунок 5 – Механіка бігу підтюпцем

- **Присутня циклічність.**
- Помірна довжина кроку.
- Велика висота кроку.
- Торкання ступи з перекатом.
- Тривале торкання.
- **Висока частота кроків.**
- Під невеликим нахилом положення корпусу.
- Помірна амплітуда руки.

6

Рисунок Б.6 – Слайд 6



7

Рисунок Б.7 – Слайд 7

Вибір програмних засобів

Таблиця 1 – Програмні засоби

Галузь	Програмний засіб
Мова розробки	Kotlin, Python
Зберігання даних	Room
Впровадження залежностей	Koin
Джерело кроків	Android cerwis
Метод фільтрації даних	Низькочастотний фільтр, фільтр піків, свої імплементації
Метод класифікації	K-найближчих сусідів (KNN)

8

Рисунок Б.8 – Слайд 8

Перетворення даних

Для перетворення даних, поданих окремо для кожної осі X, Y і Z, в єдину величину, потрібно виконати такі кроки:

- використати формулу евклідової норми (1);
- усунути вплив гравітації із загальної величини (2).

Результатом є графік (рис. 8).

$$r\text{mag} = \sqrt{x^2 + y^2 + z^2} \quad (1)$$

де $r\text{mag}$ – загальна величина прискорення з гравітацією;
 x – прискорення за координатою X;
 y – прискорення за координатою Y;
 z – прискорення за координатою Z.

$$\text{mag} = r\text{mag} - \frac{\sum_{i=0}^n r\text{mag}_i}{n} \quad (2)$$

де mag – загальна величина прискорення;
 $r\text{mag}$ – загальна величина прискорення з гравітацією;
 n – розмір масиву.

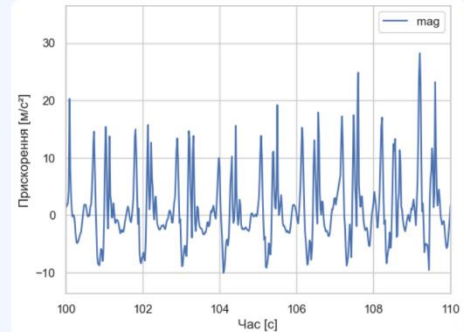


Рисунок 8 – Графік з загальною величиною лінійного прискорення

9

Рисунок Б.9 – Слайд 9

Ознаки кроків

Вигляд ознак кроків (рис. 9):

- прискорення найвищого піку;
- інтервал часу між найнижчою западиною і найвищим піком;
- інтервал часу між двома повними циклами.

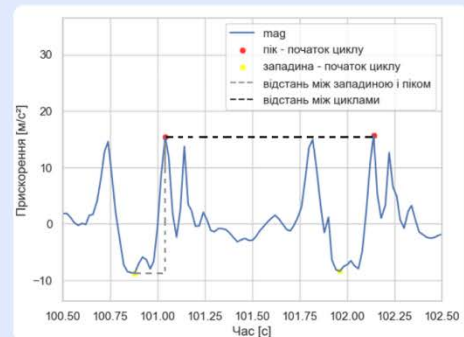


Рисунок 9 – Ознаки кроку

JetStep

10

Рисунок Б.10 – Слайд 10

Фільтрація даних

Для фільтрації даних, потрібно виконати такі кроки:

- фільтрація шумів;
- фільтрація вершин та западин (рис. 10);
- фільтрація локальних вершин та западин (рис. 11);
- знаходження істинної вершини та западини (рис. 12).

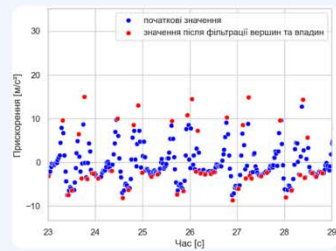


Рисунок 10 – Графік з загальною величиною лінійного прискорення

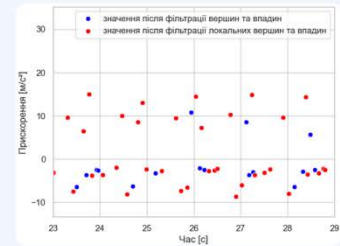


Рисунок 11 – Графік з порівняння даних до фільтрації локальних вершин та западин та після

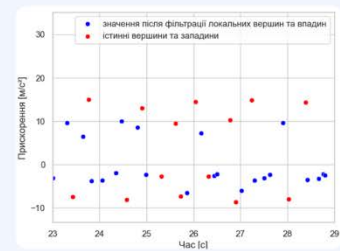


Рисунок 12 – Графік з порівнянням даних до істинних вершин та западин та після

11

Рисунок Б.11 – Слайд 11

Класифікація рухів

Для класифікації даних, потрібно виконати такі кроки:

- масштабування даних;
- класифікація даних за допомогою KNN;
- підбір оптимальної кількості сусідів.

Візуалізація ознак (рис. 13, 14)

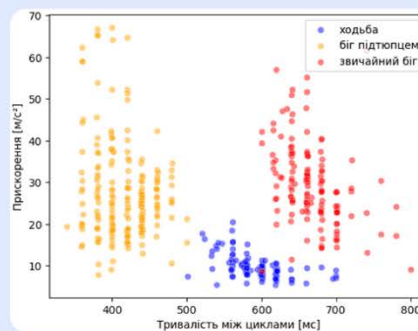


Рисунок 13 – Візуалізація ознак прискорення і тривалості між циклами

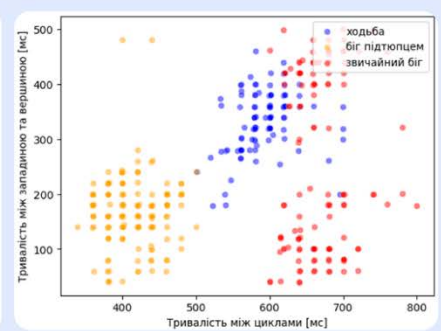


Рисунок 14 – Візуалізація ознак тривалості між западиною і вершиною та тривалості між циклами

12

Рисунок Б.12 – Слайд 12

Архітектура і структура додатка

Застосунок реалізовано відповідно до принципів Чистої Архітектури (Clean Architecture), що передбачає розбиття проєкту на модулі згідно з рівнями відповідальності (рис. 15).

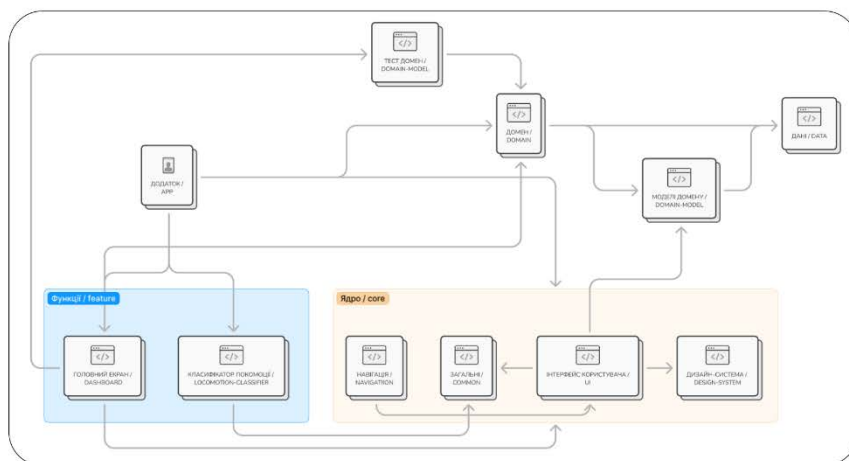


Рисунок 15 – Структура модулів

13

Рисунок Б.13 – Слайд 13

Інтерфейс головного застосунку «JetStep»

Екран «Головний» має лише одну функцію – показувати користувачеві актуальну інформацію про кроки (рис. 16).

Модальне вікно «Ручне додавання кроків» дозволяє вручну додавати кроки з зазначенням їх типу за будь-який проміжок часу (рис. 17).



Рисунок 16 – Екран «Головний»

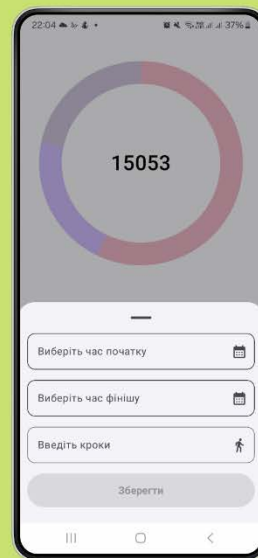


Рисунок 17 – Модальне вікно «Ручне додавання кроків»

14

Рисунок Б.14 – Слайд 14

JetStep	Тестування та результати		
	Таблиця 2 – Таблиця розпізнавання під час ходьби		
		5 кроків	30 кроків
		100 кроків	
JetStep	Перша ітерація	–	Біг підтюпцем
	Друга ітерація	–	Біг підтюпцем
	Третя ітерація	–	Ходьба
	Ходьба	Ходьба	Ходьба
JetStep	Таблиця 3 – Таблиця розпізнавання під бігу підтюпцем		
		5 кроків	30 кроків
		100 кроків	
	Перша ітерація	–	Біг підтюпцем
JetStep	Друга ітерація	–	Біг підтюпцем
	Третя ітерація	–	Звичайний біг
	Звичайний біг	Звичайний біг	Звичайний біг
	Біг підтюпцем	Біг підтюпцем	Біг підтюпцем
JetStep	Таблиця 4 – Таблиця розпізнавання під час бігу		
		5 кроків	30 кроків
		100 кроків	
	Перша ітерація	–	Біг підтюпцем
JetStep	Друга ітерація	–	Звичайний біг
	Третя ітерація	–	Звичайний біг
	Звичайний біг	Звичайний біг	Звичайний біг
	Біг підтюпцем	Біг підтюпцем	Біг підтюпцем
15			

Рисунок Б.15 – Слайд 15

JetStep	Висновки		
	У рамках магістерської дипломної роботи було розроблено нативний Android-додаток JetStep, призначений для моніторингу фізичної активності користувача. Перед початком розробки було проведено аналіз вимог до застосунку, вивчено існуючі аналоги, а також специфіку рухів. Окрім цього, виконано порівняння наявних бібліотек, середовищ розробки та мов програмування. Розробка Android-додатка здійснювалася в середовищі Android Studio, тоді як класифікатор було розроблено в Jupyter Notebook. У процесі роботи дотримувалися принципів Чистої Архітектури, реалізовано архітектурний патерн MVI. Завдяки цьому вдалося створити стабільну, менш схильну до багів і легше підтримувану систему. Для класифікації та фільтрації даних використовувалися стандартні методи, такі як фільтр нижніх частот і алгоритм К-найближчих сусідів, а також було розроблено декілька власних алгоритмів, адаптованих до специфіки завдання.		
	Дякую за увагу!		
	16		

Рисунок Б.16 – Слайд 16