НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Застосунок для розумного годинника для моніторингу активності

Виконав студент IV курсу, групи IT-92
(шифр групи)
Щербаков Антон Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Ліхоузова Т. А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викладач, Вітковська І. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Солдатова М. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Щербакову Антону Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Застосунок для розумного годинника для моніторингу активності

керівник проєкту Ліхоузова Тетяна Анатоліївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 31 » травня 2023 р. № 2101-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Моделювання та конструювання програмного забезпечення: опис структури бази даних, опис архітектури застосунку.
3) Аналіз якості та тестування програмного забезпечення.
4) Впровадження та супровід програмного забезпечення: розгортання та підтримка програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань
2) Схема бази даних
3) Креслення вигляду екранних форм
4) Схема структурна компонентів програмного забезпечення

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «10» березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літератури за тематикою проєкту	21.02.2023	
2	Розробка технічного завдання	03.03.2023	
3	Аналіз вимог та уточнення специфікацій	19.03.2023	
4	Проєктування структури програмного забезпечення, проєктування компонентів	30.03.2023	
5	Програмна реалізація програмного забезпечення	05.04.2023	
6	Тестування програмного забезпечення	10.04.2023	
7	Розробка матеріалів текстової частини проєкту	14.04.2023	
8	Розробка матеріалів графічної частини проєкту	20.04.2023	
9	Оформлення технічної документації проєкту	29.04.2023	
10	Подання ДП на попередній захист	02.06.2023	
11	Подання ДП рецензенту	12.06.2023	
12	Подання ДП на основний захист	17.06.2023	

Студент _____ Антон ЩЕРБАКОВ
(підпис) (ініціали, прізвище)

Керівник _____ Тетяна ЛІХОУЗОВА
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 45 таблиць, 39 рисунків та 8 джерел – загалом 74 сторінки.

Дипломний проєкт присвячений розробці фітнес застосунку для розумного годинника для моніторингу активності. Особливістю застосунку є можливість аналізувати поточне тренування і на його основі надавати поради користувачу.

Мета: покращити аналіз зібраної статистики в додатку для розумних годинників та надавати рекомендації стосовно продовження тренування на її основі.

Об'єкт дослідження: здоров’я та фітнес.

Предмет дослідження: застосунок для аналізу фізичної активності.

У першому розділі «Аналіз вимог до програмного забезпечення» розглядається загальний опис розумних годинників та робиться порівняння наявних аналогів, їх переваг та недоліків. Також у цьому розділі наводиться діаграма варіантів використання і на її основі описуються функціональні вимоги

Другий розділ «Моделювання та конструювання програмного забезпечення» присвячений аналізу наявних бізнес процесів і деталізації їх завдяки BPMN діаграмам. Також у цьому розділі було описано архітектуру застосунку.

У третьому розділі було виконано оцінку якості написаного коду та проведено мануальне тестування для усіх наявних в застосунку функцій.

В четвертому розділі було описано процес завантаження застосунку в магазин застосунків App Store та надано інформацію стосовно заповнення усіх необхідних для цього форм та документів.

Результати роботи пройшли апробацію на міжнародній науково практичній конференції «Наука, освіта та технології: актуальні проблеми теорії та практики» та опубліковані в матеріалах конференції.

КЛЮЧОВІ СЛОВА: WEARABLE DEVICES, XCODE, WATCHOS, HEALTH AND FITNESS, HEALTHKIT, БАЗА ДАНИХ

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 45 tables, 39 figures and 8 sources - a total of 74 pages.

The diploma project is devoted to the development of a fitness application for a smart watch for activity monitoring. A feature of the application is the ability to analyze the current workout and provide advice to the user based on it.

The goal: a brief analysis of the collected statistics in the application for the daily draw and to give recommendations regarding the continuation of training on its basis.

Research object: health and fitness.

Research subject: application for physical activity analysis.In the first section, "Analysis of software requirements", a general description of smart watches is considered and a comparison of existing analogues, their advantages and disadvantages is made. This section also provides a use case diagram and describes the functional requirements based on it

The second section "Software modeling and design" is devoted to the analysis of existing business processes and their detailing thanks to BPMN diagrams. The application architecture was also described in this section.

In the third section, the quality assessment of the written code was performed and manual testing was carried out for all functions available in the application.

The fourth chapter described the process of uploading an application to the App Store and provided information on filling out all the necessary forms and documents.

The results of the work were tested at the international scientific and operational conference "Science, research and technology: actual problems of theory and practice" and published in the conference materials.

KEYWORDS: WEARABLE DEVICES, XCODE, WATCHOS, HEALTH AND FITNESS, HEALTHKIT, DATABASE

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9228.045430.01.91	Технічне завдання	20	
3	А4	КПІ.ІТ-9228.045430.02.81	Пояснювальна записка	74	
4	А4	КПІ.ІТ-9228.045430.03.12	Текст програми	21	
5	А4	КПІ.ІТ-9228.045430.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9228.045430.05.34	Керівництво користувача	14	
7	А3	КПІ.ІТ-9228.045430.06.99	Графічний матеріал	4	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-9228.045430.00.90		
Розроб.		Щербаков А.О.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Ліхоузова Т.А.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-92	
Н.контр.		Вітковська І.І.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2023 р.

ЗАСТОСУНОК ДЛЯ РОЗУМНОГО ГОДИННИКА ДЛЯ МОНІТОРИНГУ АКТИВНОСТІ

Технічне завдання

КПІ.ІТ-9228.045430.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Антон ЩЕРБАКОВ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: ЗАСТОСУНОК ДЛЯ РОЗУМНОГО ГОДИННИКА ДЛЯ МОНІТОРИНГУ АКТИВНОСТІ.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення «ЗАСТОСУНОК ДЛЯ РОЗУМНОГО ГОДИННИКА ДЛЯ МОНІТОРИНГУ АКТИВНОСТІ» КПІ.ІТ-9228.045430.01.91, котрий використовується для моніторингу фізичних параметрів та призначений для аналізу поточного тренування спортсмена та формування порад на основі зібраних та проаналізованих даних.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки «ЗАСТОСУНОК ДЛЯ РОЗУМНОГО ГОДИННИКА ДЛЯ МОНІТОРИНГУ АКТИВНОСТІ» є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для моніторингу фізичних параметрів спортсмена та надання йому порад та застережень, сформованих на основі даних поточного тренування.

Метою розробки є створення зручного додатку для розумних годинників, який буде аналізувати зібрану статистику тренувань та на її основі давати рекомендації по проведенню поточного тренування.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– При першому запуску застосунку має відображатися запит на надання доступу до даних здоров’я Health – рисунок 4.1

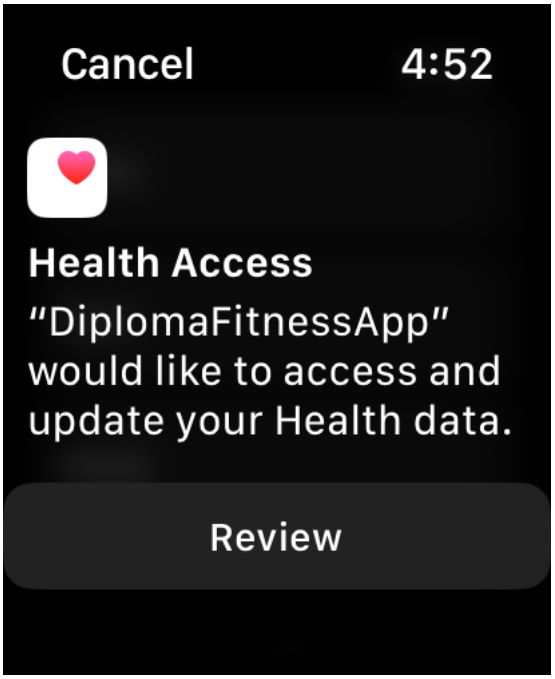


Рисунок 4.1 – Запит застосунку на доступ до даних

– При натисканні кнопки «Review» на екрані запиту відбувається перехід до екрану з наданням доступу на запис даних - рисунок 4.2

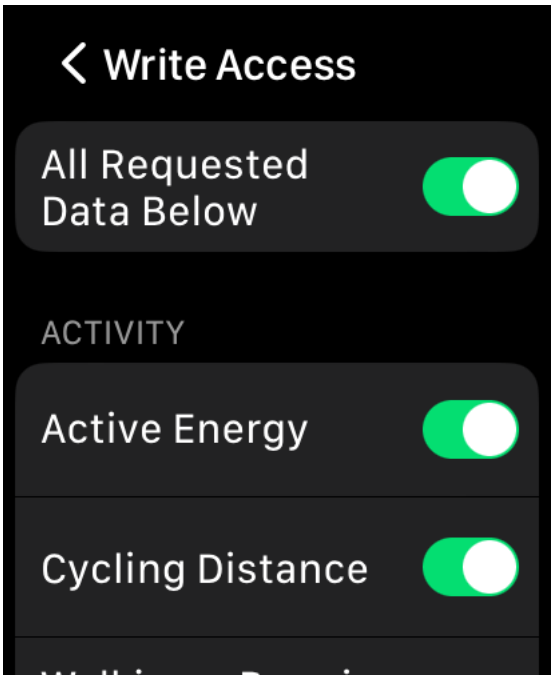


Рисунок 4.2 – Надання доступу на запис даних.

– При натисканні кнопки «Ok» на екрані запису даних відбувається перехід до екрану з наданням доступу на читання даних – рисунок 4.3

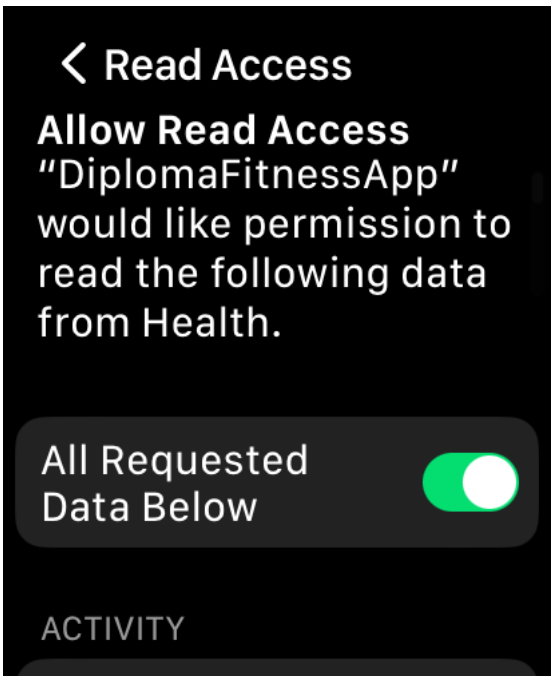


Рисунок 4.3 – Надання доступу на читання даних

– Через те, що це перша взаємодія користувача з застосунком, перед початком тренування необхідно встановити налаштування користувача. Це можна зробити, перейшовши в меню налаштувань лівим свайпом – рисунок 4.4

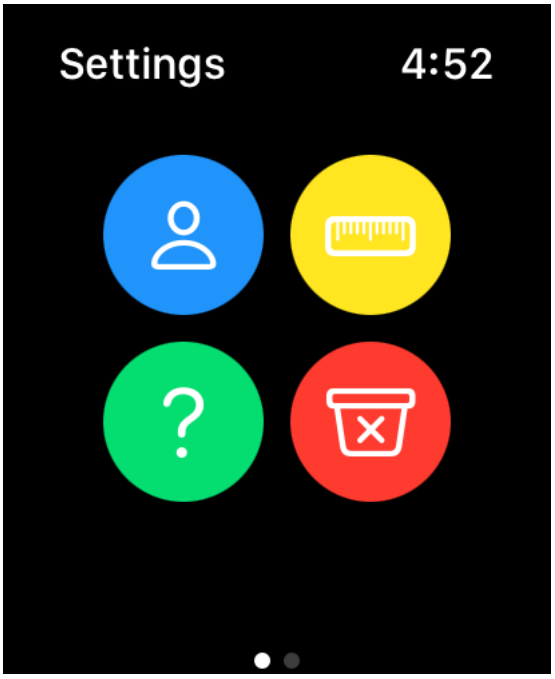


Рисунок 4.4 – Меню налаштувань застосунку

– При натисканні синьої кнопки з іконкою людини відкривається меню налаштування віку – рисунок 4.5

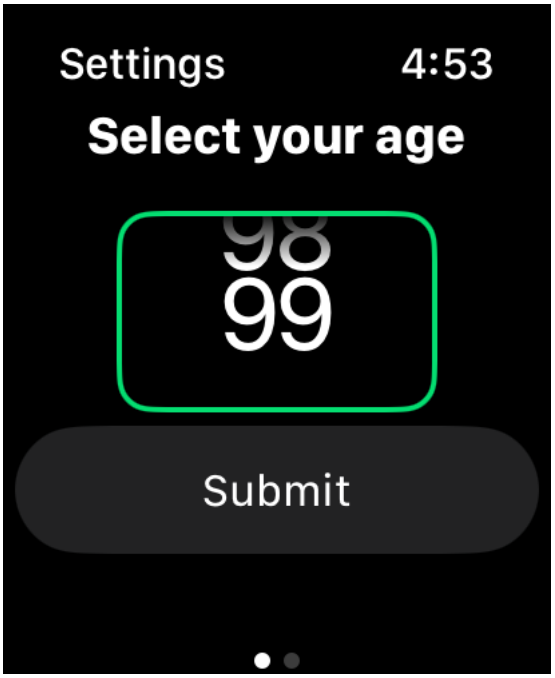


Рисунок 4.5 – Меню налаштування віку користувача

– При натисканні кнопки «Submit» відбувається перехід до меню налаштувань – рисунок 4.4

– При натисканні жовтої кнопки з іконкою лінійки відкривається меню налаштування дистанції – рисунок 4.6.

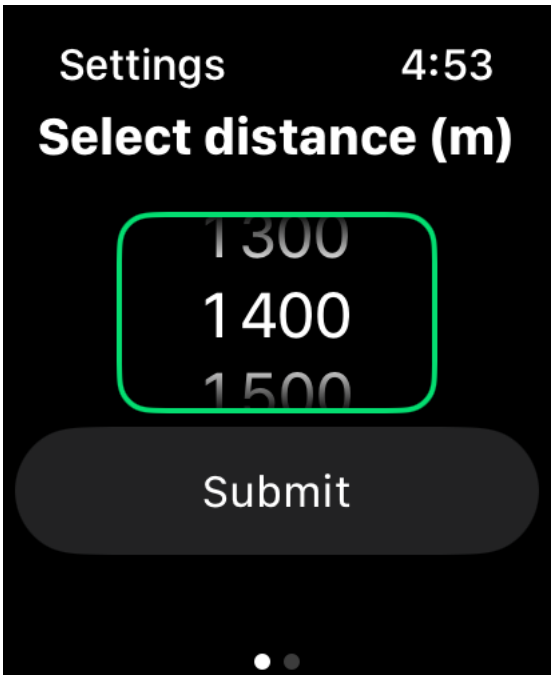


Рисунок 4.6 – Меню налаштування дистанції користувача

– При натисканні кнопки «Submit» відбувається перехід до меню налаштувань – рисунок 4.4

– При натисканні зеленої кнопки з іконкою знака питання відкривається меню отримання довідки – рисунок 4.7

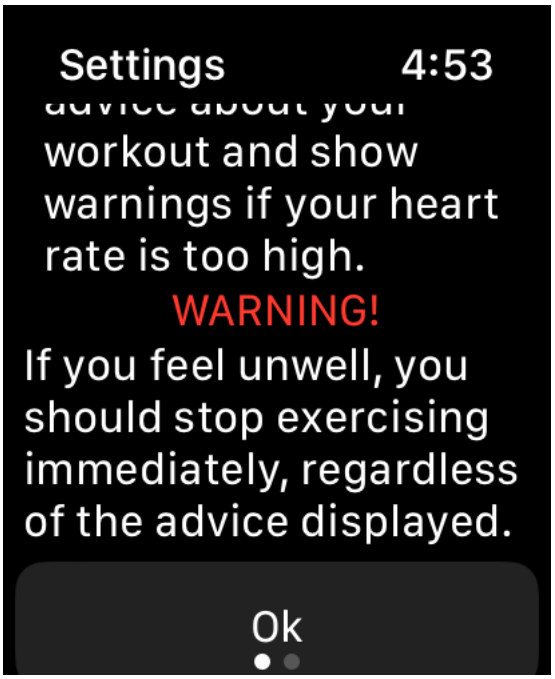


Рисунок 4.7 – Меню довідки з короткою інформацією про застосунок

– При натисканні кнопки «Ok» відбувається перехід до меню налаштувань – рисунок 4.4

– При натисканні червоної кнопки з іконкою корзини відкривається меню видалення даних– рисунок 4.8

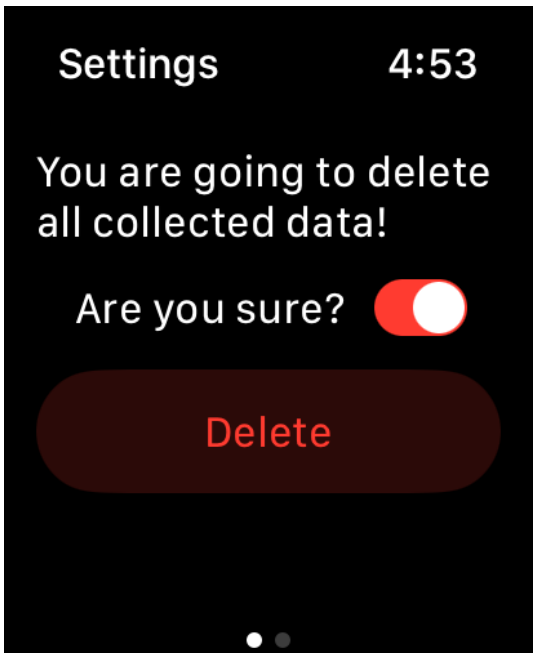


Рисунок 4.8 – Меню видалення даних тренувань з підтвердженням видалення

– При натисканні кнопки «Delete» відбувається перехід до меню налаштувань – рисунок 4.4

– При правому свайпі відбувається перехід до головного меню вибору тренувань – рисунок 4.9

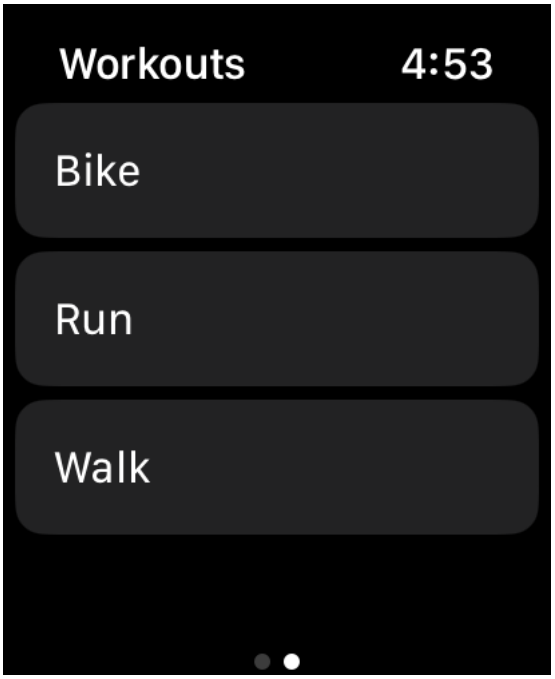


Рисунок 4.9 – Головне меню з наявними тренуваннями

– При виборі одного тренування відкривається екран з даними фізичної активності поточного тренування – рисунок 4.10

Рисунок 4.10 – Меню відображення даних поточного тренування

– При свайпі вниз відкривається екран з порадами та застереженнями – рисунок 4.11 – 4.12

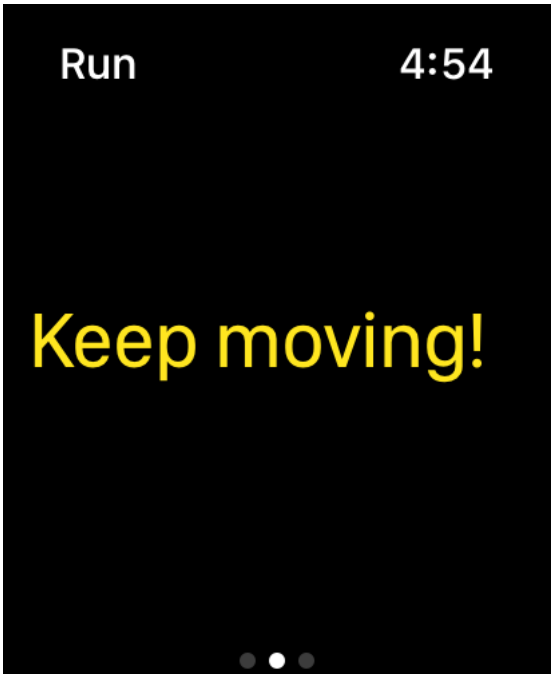


Рисунок 3.11 – Екран порад користувачу

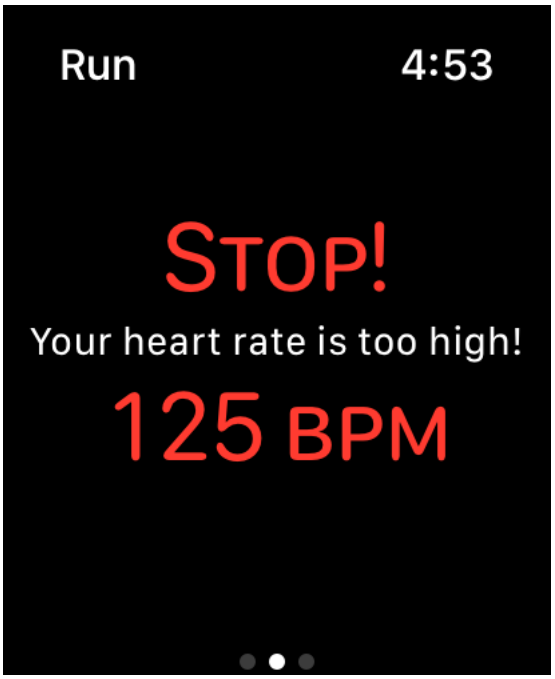


Рисунок 4.12 – Попередження користувача про досягнення максимального рекомендованого пульсу

– При свайпі вправо відкривається екран керування музикою – рисунок 3.13

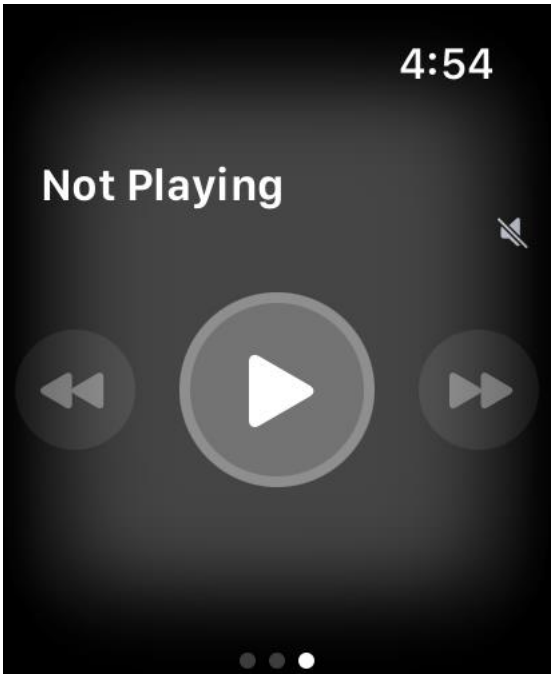


Рисунок 3.13 – Меню керування музикою

– При подвійному свайпі вліво з меню керування музикою відкривається меню керування тренуванням - рисунок 3.14

Рисунок 3.14 – Меню керування тренуванням

– При натисканні кнопки «Pause» пауза

– При натисканні кнопки «Resume» тренування продовжується – рисунок 3.15

Рисунок 3.15 – Зняття паузи та продовження тренування

– При натисканні кнопки «End» відкривається екран зі зведеною інформацію по тренуванню – рисунок 3.16

Рисунок 3.16 – Зведена інформація стосовно проведеного тренування

4.1.2 Для користувача:

– Забезпечити можливість встановлення віку користувача.

– Забезпечити можливість встановлення дистанції наступного тренування.

4.1.3 Для адміністратора системи (якщо він передбачений):

Адміністратора у системі не передбачено

4.1.4 Додаткові вимоги:

– Забезпечити можливість синхронізувати дані виконаного тренування з застосунком Fitness на смартфоні користувача.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно ДСанПін 3.3.2.007 – 98.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на WatchOS-сумісних розумних годинниках.

Мінімальна конфігурація технічних засобів:

- Назва пристрою Apple Watch 4;
- Об‘єм вбудованої пам‘яті: 50MB;
- об‘єм ОЗП: 1 Гб;

Рекомендована конфігурація технічних засобів:

- Назва пристрою Apple Watch 6;
- Об‘єм вбудованої пам‘яті: 50MB;
- об‘єм ОЗП: 1.5 Гб.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WatchOS.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі:

- Вік користувача;
- Запланована дистанція тренування.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі:

– Текстові поради стосовно подальших дій в поточному тренуванні;

– Текстове застереження про перевищення максимального рекомендованого пульсу.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування SWIFT версії 5.0

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі XCode

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді окремих файлів з розширенням .swift

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати файл з застосунком, готовий до завантаження у App Store.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема структурна компонентів програмного забезпечення;
- схема бази даних;
- креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проєкту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.03	Специфікації програмного забезпечення
4.	Проєктування структури програмного забезпечення, проєктування компонентів	30.03	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.04	Тести, результати тестування
7.	Розробка матеріалів текстової частини проєкту	14.04	Пояснювальна записка
8.	Розробка матеріалів графічної частини проєкту	20.04	Графічний матеріал проєкту
9.	Оформлення технічної документації проєкту	29.04	Технічна документація
10.			
11.			
12.			
13.			

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Застосунок для розумного годинника для моніторингу активності.

КПІ.ІТ-9228.045430.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
ER	–	Entity-Relation diagram
ОС	–	Операційна система.
БД	–	База даних.
WatchOS	–	Операційна система для розумних годинників від компанії Apple
App Store	–	Цифровий магазин застосунків для пристроїв компанії Apple

Вступ

Зараз, коли смарт-годинники та фітнес-браслети стали не просто модним аксесуаром, а невід'ємною частиною повсякденного життя багатьох людей, з'являється все більше пристроїв з різноманітними функціями. Більшість з них можуть здійснювати моніторинг тренувань, збираючи інформацію з різних датчиків, таких як GPS, пульсометр, акселерометр. Однак, не всі такі пристрої можуть надавати користувачеві докладну аналітику та рекомендації з покращення тренування, що є важливим для досягнення успіху в спорті. Однією з ключових проблем фітнес-застосунків для смарт годинників є недостатня обробка та аналіз зібраних даних безпосередньо під час тренування. У більшості випадків користувачі самостійно повинні оцінювати показники та здійснювати аналіз даних, що може бути складним для новачків у спорті. Наприклад, правильна оцінка рекомендованого максимально допустимого пульсу для кожної людини є важливою складовою ефективного та безпечного тренування. Недотримання цієї межі може призвести до поганого самопочуття, запаморочення та навіть до серйозних проблем зі здоров'ям. На жаль, жоден проаналізований застосунок, не надає достатньої інформації щодо правильного виконання тренувань та збору показників, що є недоліком для користувачів, які хочуть займатися спортом без ризиків для свого здоров'я.

Метою цього дипломного проекту є покращення аналізу зібраної статистики в додатку для розумних годинників та надавання рекомендацій стосовно продовження тренування на її основі.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Розумні годинники, також відомі як "смарт-годинники" – це електронні пристрої, які поєднують в собі дизайн звичайних наручних годинників з рядом додаткових можливостей, таких як відображення повідомлень, відстеження фізичної активності та контролю здоров'я.

Розумні годинники можуть бути синхронізовані зі смартфонами та іншими пристроями, що дозволяє користувачам отримувати вхідні дзвінки та повідомлення безпосередньо на своєму годиннику, а інколи навіть відповідати на них завдяки ньому. Також більшість розумних годинників мають вбудований GPS-чіп, який дозволяє відстежувати місцезнаходження користувача та записувати його фізичну активність під час повсякденних справ та фізичних тренувань. Основною ж цінністю розумних годинників є можливість контролювати параметри свого організму, такі як вимірювання кров'яного тиску, рівня кисню в крові та електрокардіограми [1].

Загалом, розумні годинники є корисними та зручними пристроями, які дозволяють користувачам отримувати повідомлення, відстежувати фізичну активність та контролювати своє здоров'я. Завдяки постійному розвитку технологій, вже наявна функціональність розумних годинників постійно покращується, що дуже актуально у контексті точності роботи датчиків моніторингу здоров’я.

1.2 Змістовний опис і аналіз предметної області

Розумні годинники – це невеликі комп’ютери, які за технічним наповненням подібні до смартфонів. Вони пропонують низку функцій, частина з яких є унікальною для цього типу пристроїв і стосуватися аналізу здоров’я та фізичних параметрів користувача. Інші доповнюють вже існуючу

функціональність смартфону чи комп’ютера користувача. У цьому випадку розумний годинник займається тільки збором та пересиланням даних на інший пристрій користувача, який вже робить усю необхідну обробку.

Однією з головних проблем розробки програмного забезпечення для розумних годинників є відсутність багатьох можливостей для монетизації цього застосунку. Ця проблеми веде до відсутності інтересу великих компаній та команд до цього виду програмного забезпечення. Відтак, основною рушійною силою всіх розробок є або ентузіасти, або компанії – виробники своїх смарт годинників, які більше зацікавлені в розробці базового набору програм, ніж до наповнення свої магазинів з застосунками.

Також проблемами є невеликий розмір екрану та обмежена обчислювальна потужність, порівняно з настільними або мобільними пристроями. Це означає, що розробники повинні оптимізувати як користувацький інтерфейс, так і алгоритми свого програмного продукту для підвищення швидкодії застосунку.

Іншою проблемою є різноманітність ринку розумних годинників, де різні пристрої працюють на різних операційних системах і мають різні апаратні характеристики та операційні системи. Розробникам необхідно враховувати особливості та обмеження кожного пристрою та платформи та переконатися, що їхні програми оптимізовані для кожного з них, або зосередитись на створенні своїх програмних продуктів тільки для обмеженого ряду пристроїв.

Через відсутність інструментів для розробки застосунків одразу для декількох платформ, у цій роботі було обрано шлях розробки застосунку тільки для однієї платформи – для WatchOS. Також однією з цілей цього дипломного проєкту є створення повністю автономного від смартфону застосунку. Для цього буде розроблений простий та мінімалістичний інтерфейс та простий алгоритм для аналізу даних тренування.

1.3 Аналіз існуючих технологій та успішних ІТ-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації застосунку для розумного годинника для моніторингу активності. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Одним з ключових алгоритмів цього дипломного проєкту буде алгоритм формування порад користувачу стосовно поточного тренування та його продовження.

Перед початком тренування користувач має встановити цільову дистанцію у метрах. Для аналізу витрат енергії на досягнення цілі, алгоритм буде прогнозувати витрачену енергію у кілокалорія на основі даних поточного тренування. Через те, що дані і витрат енергії і дистанції поступово збільшуються, а спортсмени завжди намагаються тримати рівний темп або плавно його змінювати, для прогнозування витрат енергії добре підійде алгоритм лінійної регресії. Після отримання прогнозованого значення, буде виконано порівняння цього значення з вже наявними значеннями по раніше проведеним тренуванням. Наприклад, якщо раніше на 5км дистанції спортсмен витрачав 200 кілокалорій, а зараз за прогнозом витратить 300, застосунок йому порадить зменшити темп для досягнення цілі.

Також можлива ситуація, коли встановлена користувачем ціль буде більше, ніж максимальна дистанція раніше проведених тренувань. В такому разі застосунок прорахує витрату калорій для вже відомої дистанції і буде показувати поради, орієнтуючись на це число, але з корегуванням, що реальна запланована дистанція буде більше.

Застосунок також буде попереджати про досягнення максимального рекомендованого пульсу. Максимальний рекомендований пульс можливо порахувати з різниці числа 220 та віку користувача. Хоча кожен організм має

індивідуальні показники, дослідження показують лінійну залежність між віком та максимальним пульсом людини. Слід зазначити, що дослідник не радять спиратися на цю формулу, рахуючи максимальний пульс дітей до 10 років [2].

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Єдиним доступним варіантом IDE для розробки застосунків для Apple watch є XCode.

XCode - IDE для розробки програмного забезпечення для iOS, iPadOS, macOS та watchOS. XCode має інструменти для розробки та тестування застосунків.

Існує 2 мови програмування, за допомогою яких можна розробляти застосунки для Apple Watch – Swift та ObjectiveC. Мова Objective-C давно використовується для розробки застосунків для платформ компанії Apple і все ще підтримується компанією. Однак, останнім часом компанія підштовхує розробників до використання мови програмування Swift, яка є більш сучасною мовою, призначеною для роботи з їхніми фреймворками та розробки застосунків в єдиній стилістиці та з єдиною кодовою базою.

Таким чином, незважаючи на те, що все ще можна створювати програми Apple Watch за допомогою мови Objective-C, навіть сама компанія Apple рекомендує використовувати більш сучасну мову Swift для розробки нових застосунків та подальшої їх підтримки

Для розробки користувацького інтерфейсу існує два популярних фреймворки – SwiftUI та UIKit. У цьому проєкті буде використано SwiftUI, бо він пропонує кілька переваг перед UIKit, таких як:

- Можливість динамічного попереднього перегляду сторінок. SwiftUI має функцію попереднього перегляду сторінок в реальному часі, яка дозволяє розробникам бачити зміни в інтерфейсі користувача в режимі

реального часу, без необхідності кожен раз знову збирати застосунок. Це допомагає скоротити час розробки та збільшити продуктивність.

- Необхідність написання меншої кількості коду. SwiftUI вимагає менше коду для створення компонентів інтерфейсу користувача, ніж UIKit. Це досягається завдяки тому, що SwiftUI має кілька вбудованих елементів керування, які можна легко налаштувати за допомогою лише кількох рядків коду.

Загалом SwiftUI пропонує більш сучасний і спрощений підхід до створення компонентів інтерфейсу користувача, ніж UIKit, що робить його чудовим вибором для розробки фітнес застосунку для Apple Watch [3].

Для роботи з даними, які відносяться до фізичних характеристик та здоров’я користувача, компанія Apple пропонує використовувати фреймворк HealthKit. HealthKit надає розробникам набір інструментів для безпечного зберігання, керування та отримання даних про здоров’я та фізичну форму користувача. Це включає дані про фізичну активність, харчування, сон, пульс та артеріальний тиск. Збираючи та аналізуючи ці дані, користувачі можуть отримати уявлення про своє здоров’я та самопочуття.

Окрім того, що HealthKit дозволяє користувачам переглядати дані про своє здоров’я в одному місці, HealthKit також дозволяє користувачам ділитися своїми даними зі сторонніми застосунками для аналізу фізичної активності [4].

1.3.3 Аналіз відомих програмних продуктів

Для Apple Watch доступно багато програм для фітнесу, кожна з яких має свої унікальні функції та особливості. Наведемо деякі популярні програми для фітнесу та їх особливості:

Apple Fitness: власна програма для фітнесу від Apple, розроблена спеціально для Apple Watch. Застосунок пропонує широкий спектр тренувань, включаючи біг, йогу, їзду на велосипеді та танці, а також показує показники

фізичної активності. Головну сторінку застосунку в магазині застосунків можна побачити на рисунку 1.1.

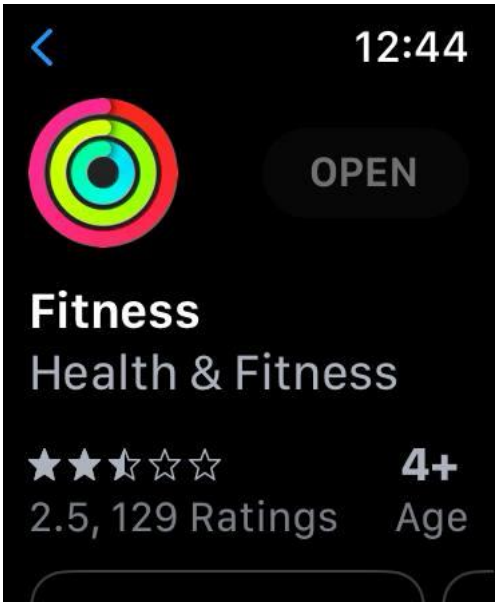


Рисунок 1.1 – Сторінка застосунку Apple Fitness

MyFitnessPal - популярна програма для відстеження калорій і фізичних вправ. MyFitnessPal інтегрується з Apple Watch, щоб відстежувати щоденні кроки та тренування. Застосунок також пропонує харчовий щоденник, де ви можете записувати свої прийоми їжі та відстежувати споживання калорій. Головну сторінку застосунку в магазині застосунків можна побачити на рисунку 1.2.

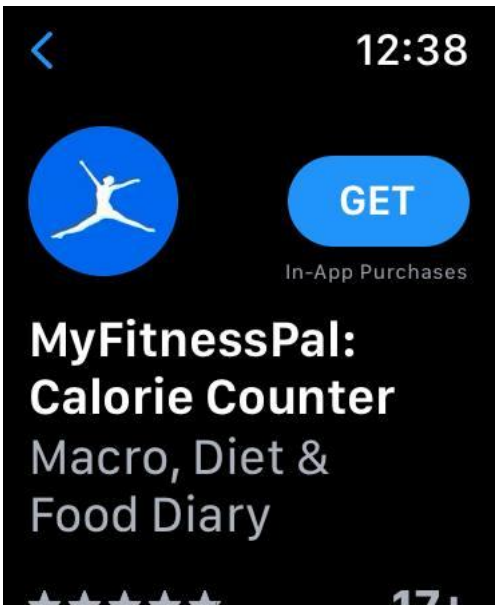


Рисунок 1.2 – Сторінка застосунку MyFitnessPal

Strava - застосунок для соціального фітнесу, який дозволяє вам відстежувати свої тренування та ділитися ними з друзями. Він також пропонує виклики та змагання, які допоможуть мотивувати вас досягти ваших фітнес-цілей. Головну сторінку застосунку в магазині застосунків можна побачити на рисунку 1.3.

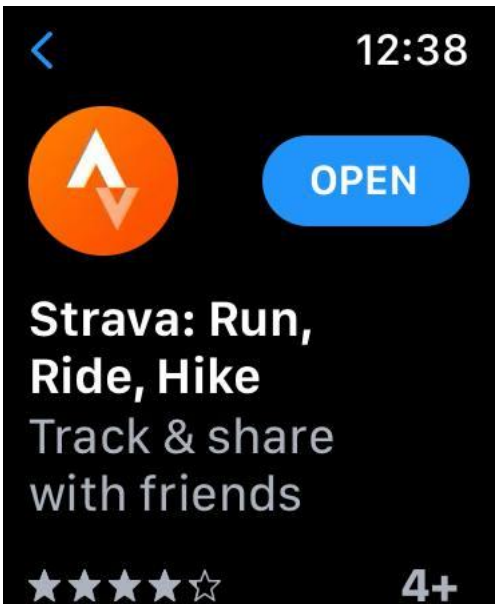


Рисунок 1.3 – Сторінка застосунку Strava

Nike Training Club - фітнес-застосунок від Nike, який пропонує різноманітні тренування, зокрема силові вправи, циклічні та йогу. Він надає персоналізовані плани тренувань і дозволяє відстежувати ваш прогрес. Головну сторінку застосунку можна побачити на рисунку 1.4.

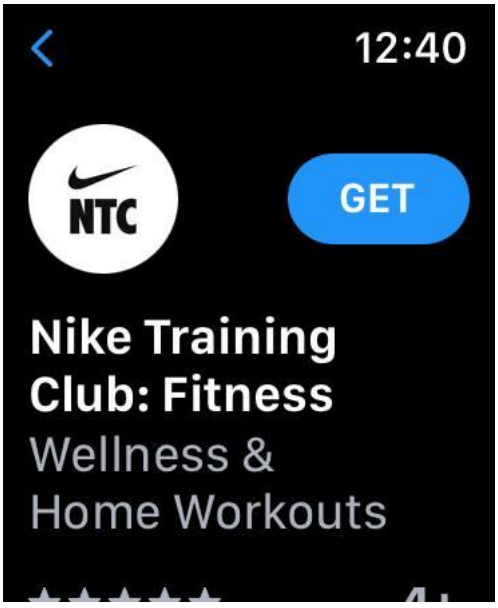


Рисунок 1.4 – Сторінка застосунку Nike Training Club

FitOn - застосунок пропонує набір вправ та програму тренувань для зменшення ваги, занять йогою та виконання повсякденних занять спортом.

Головну сторінку застосунку в магазині застосунків можна побачити на рисунку 1.5.

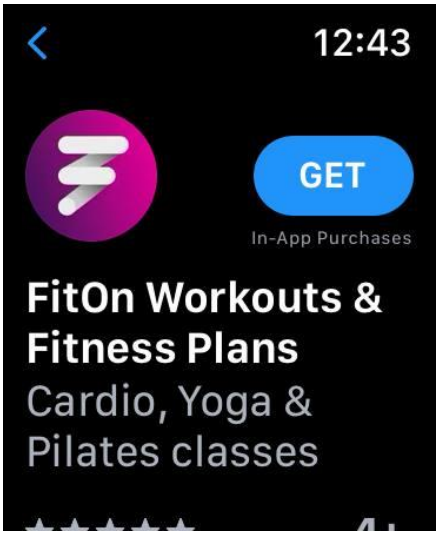


Рисунок 1.5 – Сторінка застосунку FitOn

Загалом відмінності між цими фітнес-застосунками зводяться до їхніх особливостей і типів тренувань, які вони пропонують. Деякі програми зосереджені на соціальних функціях і змаганнях, а інші пропонують персоналізовані плани тренувань або зосереджені на певних типах тренувань, як-от їзда на велосипеді чи йога.

Незважаючи на широкий асортимент застосунків для відстеження та запису тренувань користувача, жоден не аналізує дані безпосередньо у процесі тренування та не формує відповідні поради та рекомендації. Усі наведені застосунки нерозривно пов’язані з застосунком на смартфоні. Ключовою відмінністю застосунка, розроблюваного в рамках цього дипломного проєкту, є можливість аналізувати дані та формувати поради та застереження для користувача безпосередньо у процесі тренування, коли вони найбільше потрібні.

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є формування порад спортсмену під час тренування, більше функцій на рисунку 1.6.

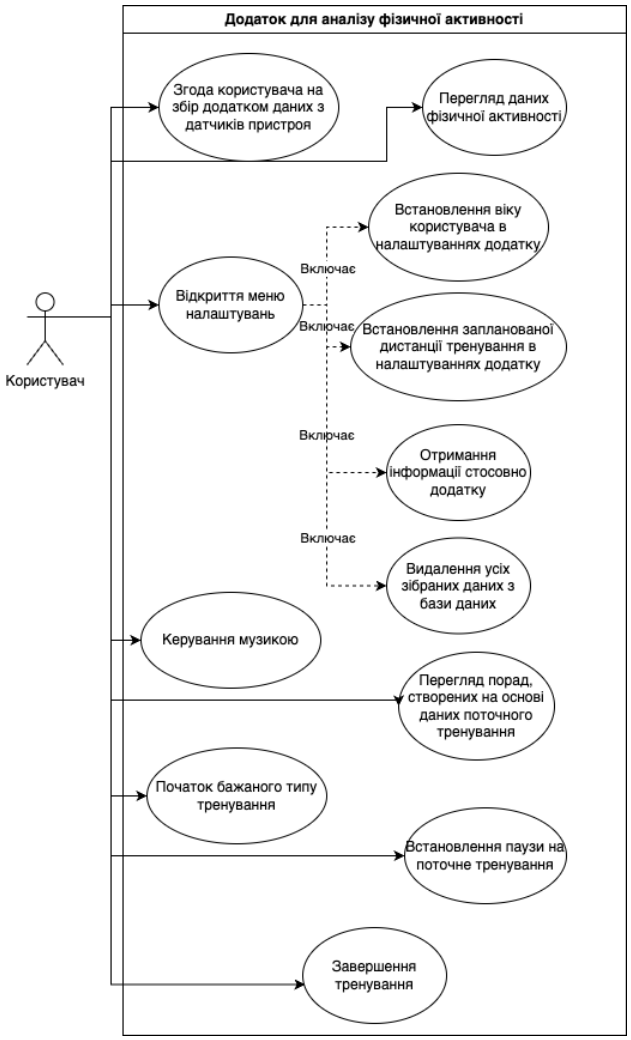


Рисунок 1.6 – Діаграма варіантів використання

В таблицях 1.1 - 1.11 наведені варіанти використання програмного забезпечення.

Таблиця 1.1 - Варіант використання UC-1

Use case name	Надання дозвіл застосунку на роботу з даними
Use case ID	UC-01
Goals	Надання доступу застосунку для читання та запис даних на пристрій
Actors	Користувач
Trigger	Користувач перший раз відкриває застосунок
Pre-conditions	-

Продовження таблиці 1.1

Flow of Events	Користувач обирає іконку застосунку серед наявних на пристрої та натискає її. Користувач бачить запит на отримання даних з датчиків пристрою, та прохання дозволити їх збір та обробку.
Extension	В випадку не надання дозволу застосунок не зможе показувати дані з датчиків
Post-Condition	Відкриття сторінки вибору тренування

Таблиця 1.2 - Варіант використання UC-2

Use case name	Відкриття меню налаштувань
Use case ID	UC-02
Goals	Надати користувачу доступ до налаштувань застосунку
Actors	Користувач
Trigger	Користувач робить свайп вліво зі стартового меню
Pre-conditions	-
Flow of Events	Користувач робить свайп вліво та переходить до меню налаштувань застосунку, де знаходяться налаштування віку, дистанції та інформація про застосунок.
Extension	-
Post-Condition	Користувач перейшов на сторінку вибору налаштувань застосунку

Таблиця 1.3 - Варіант використання UC-3

Use case name	Встановлення віку користувача
Use case ID	UC-03
Goals	Встановити вік користувача
Actors	Користувач
Trigger	Користувач натискає на іконку встановлення віку.
Pre-conditions	Відкрите меню налаштувань
Flow of Events	Користувач натискає на іконку встановлення віку в налаштуваннях. Після переходу в меню встановлення віку користувач обирає свій вік від 1 до 100 років та натискає кнопку «Submit»
Extension	У випадку не встановлення віку, система зберігає останній встановлений вік, або вік за замовчуванням (0)

Продовження таблиці 1.3

Post-Condition	Користувач повертається до меню налаштувань

Таблиця 1.4 - Варіант використання UC-4

Use case name	Встановлення дистанції тренування
Use case ID	UC-04
Goals	Встановити дистанцію тренування
Actors	Користувач
Trigger	Користувач натискає на іконку встановлення дистанції.
Pre-conditions	Відкрите меню налаштувань
Flow of Events	Користувач натискає на іконку встановлення дистанції в налаштуваннях. Після переходу в меню встановлення віку користувач обирає дистанцію та натискає кнопку «Submit»
Extension	У випадку не встановлення дистанції, система зберігає останню встановлену дистанцію, або дистанцію за замовчуванням (5 км)
Post-Condition	Користувач повертається до меню налаштувань

Таблиця 1.5 - Варіант використання UC-5

Use case name	Отримання інформації про застосунок
Use case ID	UC-05
Goals	Отримати коротку інформацію про застосунок
Actors	Користувач
Trigger	Користувач натискає на іконку отримання інформації.
Pre-conditions	Відкрите меню налаштувань
Flow of Events	Користувач натискає на іконку отримання інформації в налаштуваннях. Після переходу в меню та читання інформації користувач натискає кнопку “Submit”
Extension	-
Post-Condition	Користувач повертається до меню налаштувань

Таблиця 1.6 - Варіант використання UC-6

Use case name	Початок тренування
Use case ID	UC-06

Продовження таблиці 1.6

Goals	Почати необхідне тренування
Actors	Користувач
Trigger	Користувач натискає на кнопку з необхідним йому тренуванням
Pre-conditions	Відкрите стартове меню зі списком тренувань
Flow of Events	Користувач натискає на кнопку з необхідним йому тренуванням. Після цього тренування починається.
Extension	-
Post-Condition	-

Таблиця 1.7 - Варіант використання UC-7

Use case name	Встановлення паузи
Use case ID	UC-07
Goals	Тимчасово припинити тренування
Actors	Користувач
Trigger	Користувач натискає кнопку паузи
Pre-conditions	Відкрите меню опцій тренування
Flow of Events	Користувач свайпом вліво переходить в меню опцій тренування та натискає кнопку паузи
Extension	-
Post-Condition	-

Таблиця 1.8 - Варіант використання UC-8

Use case name	Завершення тренування
Use case ID	UC-08
Goals	Завершити тренування
Actors	Користувач
Trigger	Користувач натискає кнопку завершення тренування
Pre-conditions	Відкрите меню опцій тренування
Flow of Events	Користувач свайпом вліво переходить в меню опцій тренування та натискає кнопку завершення тренування
Extension	-

Продовження таблиці 1.8

Post-Condition	Користувач переходить в меню в меню огляду інформації по своєму тренуванню.

Таблиця 1.9 - Варіант використання UC-9

Use case name	Перегляд порад
Use case ID	UC-09
Goals	Переглянути поради стосовно поточного тренування
Actors	Користувач
Trigger	Користувач свайпом вниз переходить на екран з порадами
Pre-conditions	Відкрите меню з метриками поточного тренування
Flow of Events	Користувач свайпом вниз переходить на екран з порадами та має змогу переглядати доступні поради.
Extension	-
Post-Condition	-

Таблиця 1.10 - Варіант використання UC-10

Use case name	Керування музикою
Use case ID	UC-10
Goals	Керувати музикою, як в момент тренування грає у сторонньому застосунку
Actors	Користувач
Trigger	Користувач свайпом вправо переходить на екран з музикою
Pre-conditions	Відкрите меню з метриками поточного тренування
Flow of Events	Користувач свайпом вправо переходить на екран з музикою та має змогу керувати треками, які грають у сторонньому музикальному застосунку.
Extension	-
Post-Condition	-

Таблиця 1.11 - Варіант використання UC-11

Use case name	Видалення усіх записаних даних з бази даних
Use case ID	UC-11

Продовження таблиці 1.11

Goals	Видалити усі наявні дані тренування в базі даних
Actors	Користувач
Trigger	Користувач натискає на іконку видалення даних.
Pre-conditions	Відкрите меню налаштувань
Flow of Events	Користувач натискає на іконку видалення даних та потрапляє у відповідне меню. Після погодження користувач натискає кнопку “Delete” і дані видаляються
Extension	У випадку не погодження з видаленням даних, буде доступна лише кнопка “Cancel”, натискання якої перенаправляє користувача у головне меню, без видалення даних.
Post-Condition	Користувач повертається до меню налаштувань

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. Модель вимог у загальному вигляді наведена в таблиці 1.12

Таблиця 1.12 – Модель вимог у загальному вигляді

Номер	Назва	Код	Пріоритет	Ризик	Статус
1	Запит на доступ до сенсорів пристрою	FR-1	Високий	Невизначений	Підтверджено
2	Перехід в меню налаштувань	FR-2	Високий	Невизначений	Підтверджено
3	Перехід в меню налаштування віку	FR-3	Високий	Невизначений	Підтверджено
4	Налаштування віку	FR-4	Високий	Невизначений	Підтверджено

Продовження таблиці 1.12

5	Перехід в меню налаштування дистанції тренування	FR-5	Високий	Невизначений	Підтверджено
6	Налаштування дистанції	FR-6	Високий	Невизначений	Підтверджено
7	Перехід в меню довідки	FR-7	Низький	Невизначений	Підтверджено
8	Початок обраного тренування	FR-8	Високий	Невизначений	Підтверджено
9	Перехід в меню керування тренуванням	FR-9	Високий	Невизначений	Підтверджено
10	Встановлення паузи	FR-10	Високий	Невизначений	Підтверджено
11	Отримання порад	FR-11	Високий	Невизначений	Підтверджено
12	Завершення тренування	FR-12	Високий	Невизначений	Підтверджено
13	Керування музикою	FR-13	Низький	Невизначений	Підтверджено
14	Видалення даних з бази даних	FR-14	Середній	Невизначений	Підтверджено

На рисунку 1.7 наведено матрицю трасування вимог, а в таблицях 1.13 – 1.24 наведений опис функціональних вимог до програмного забезпечення.

Таблиця 1.13 – Функціональна вимога FR-1

Назва	Запит на доступ до сенсорів пристрою
Опис	Система повинна запросити в користувача дозвіл на читання та обробку даних з сенсорів

Таблиця 1.14 – Функціональна вимога FR-2

Назва	Перехід в меню налаштувань
Опис	Система надає користувачу можливість відкрити меню налаштувань додатку зробивши свайп вліво з головного меню

Таблиця 1.15 – Функціональна вимога FR-3

Назва	Перехід в меню налаштування віку
Опис	Система надає користувачу можливість відкрити меню налаштування віку, натиснувши кнопку вибору віку

Таблиця 1.16 – Функціональна вимога FR-4

Назва	Налаштування віку
Опис	Система надає користувачу можливість обрати його вік в меню налаштування віку, який має знаходитись у діапазоні від 0 до 99

Таблиця 1.17 – Функціональна вимога FR-5

Назва	Перехід в меню налаштування дистанції тренування
Опис	Система надає користувачу можливість відкрити меню налаштування дистанції тренування, натиснувши кнопку вибору дистанції

Таблиця 1.18 – Функціональна вимога FR-6

Назва	Налаштування дистанції
Опис	Система надає користувачу можливість обрати бажану дистанцію його тренування, обравши один з наданих варіантів від 100м до 100000м

Таблиця 1.19 – Функціональна вимога FR-7

Назва	Перехід в меню довідки
Опис	Система надає користувачу можливість отримати невеличку довідку про застосунок

Таблиця 1.20 – Функціональна вимога FR-8

Назва	Початок обраного тренування
Опис	Система надає користувачу можливість обрати одне з трьох доступних тренувань і почати його

Таблиця 1.21 – Функціональна вимога FR-9

Назва	Перехід в меню керування тренуванням
Опис	Система надає користувачу можливість відкрити меню керування тренуванням, здійснивши свайп вліво від головного екрана тренування. Користувач отримає доступ до двох кнопок – паузи та закінчення тренування

Таблиця 1.22 – Функціональна вимога FR-10

Назва	Встановлення паузи
Опис	Система надає користувачу можливість встановити паузу під час тренування. Після цього тренування перестане записуватися до моменту повторного натискання паузи користувачем

Таблиця 1.23 – Функціональна вимога FR-11

Назва	Отримання порад
Опис	Система надає користувачу можливість перейти до екрану з порадами, зробивши свайп вниз від головного меню тренування. Після цього користувач зможе бачити текстові поради, які будуть відноситись до його тренування

Таблиця 1.24 – Функціональна вимога FR-12

Назва	Завершення тренування
Опис	Система надає користувачу можливість завершити тренування, натиснувши відповідну кнопку в меню керування тренуванням. Після цього користувач побачить всю зібрану статистику стосовно його тренування

Таблиця 1.25 – Функціональна вимога FR-13

Назва	Керування музикою
Опис	Система надає користувачу можливість керувати музикою, яка грає через сторонній застосунок на розумному годиннику. Це стандартний модуль, який надається компанією Apple і через це є зручним, корисним та простим у використанні.

Таблиця 1.26 – Функціональна вимога FR-14

Назва	Видалення даних з бази даних
Опис	Система надає користувачу можливість видалити усі раніше записані дані. Це може бути актуальним, коли користувач робив велику перерву у заняттях спортом і тепер хоче, щоб алгоритм працював лише на актуальних даних.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14
UC-1	+													
UC-2		+												
UC-3			+	+										
UC-4					+	+								
UC-5							+							
UC-6								+	+					
UC-7									+					
UC-8										+		+		
UC-9									+		+			
UC-10													+	
UC-11														+

Рисунок 1.7 – Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

Метою дипломного проєкту є розробка застосунку для аналізу фізичної активності для годинників Apple Watch згідно наступних нефункціональних вимог:

- застосунок має містити інтуїтивно зрозумілий інтерфейс;
- висока швидкодія застосунку повинна забезпечувати комфортне використання під час тренування;
- інтерфейс застосунку повинен бути адаптований під використання на маленькому екрані годинника;
- усі елементи керування застосунком повинні бути у єдиному стилі.

1.5 Постановка задачі

Метою цього дипломного проекту є покращення аналізу зібраної статистики в додатку для розумних годинників та надавання рекомендацій стосовно продовження тренування на її основі.

Застосунок буде орієнтований на заняття бігом.

Декілька перших тренувань буде збиратися статистика.

Після старту активності застосунок почне виводити усю стандартну інформацію, як темп, пульс, час активності, пройдену відстань. Ці дані будуть отримуватися з сенсорів та модулів розумного годинника.

Коли базова статистика по тренуванням буде зібрана, в застосунку почнуть відображатися поради щодо продовження тренування відповідно до спеціального алгоритму.

Алгоритм буде порівнювати уже зібрані дані з даними поточного тренування та на основі цього порівняння формувати пораду.

Поради будуть мати вигляд подібний до «Продовжуйте в такому ж темпі!», «Збільште темп!», «Зменште темп!», «Занадто високий пульс! Зупиніть тренування!».

Висновки до розділу

В цьому розділі проаналізовано вимоги до програмного забезпечення застосунку для розумного годинника для моніторингу активності.

Було проведено змістовний опис і аналіз предметної області. Наведено загальну інформацію про розумні годинники, про проблеми і складнощі розробки програмного забезпечення для розумних годинників та сформовано головну ціль цього дипломного проєкту – розробка повністю автономного від смартфону застосунку для аналізу фізичної активності.

Також було проаналізовано технології, завдяки яким можна виконати поставлену задачу. Незважаючи на наявність декількох можливих мов програмування і декількох фреймворків створення користувацького інтерфейсу, компанія Apple в останній час розвиває мову програмування Swift і фреймворк користувацького інтерфейсу SwiftUI. Вибір цих рекомендованих технологій не тільки дозволить створити сучасний продукт, який буде відповідати сучасним тенденціям розробки мобільних застосунків для платформи WatchOS, а й без проблем підтримувати цей застосунок у майбутньому.

Однією з найважливіших проаналізованих тем є аналіз вже існуючих програмних продуктів. Після проведення цього аналізу було зроблено висновок, що незважаючи на вели різноманіття різних фітнес застосунків для

розумних годинників, жоден з них не пропонує аналіз даних поточного тренування безпосередньо у процесі тренування.

Додатково були проаналізовані функціональні та нефункціональні вимоги до програмного продукту і на їх основі створено діаграму варіантів використання та матрицю трасування вимог.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для кращого розуміння усіх наявних бізнес процесів застосунку, створимо BPMN діаграму для кожного процесу.

Список процесів та відповідних діаграм наступний.

– Встановлення віку користувача в налаштуваннях (рисунок 2.1);

– Встановлення дистанції тренування користувача в налаштуваннях (рисунок 2.2);

– Отримання довідки стосовно застосунку (рисунок 2.3);

– Видалення усіх записів про тренування користувача з бази даних (рисунок 2.4);

– Виконання тренування користувачем (рисунок 2.5);

– Встановлення паузи на поточне тренування (рисунок 2.6);

– Завершення поточного тренування (рисунок 2.7).

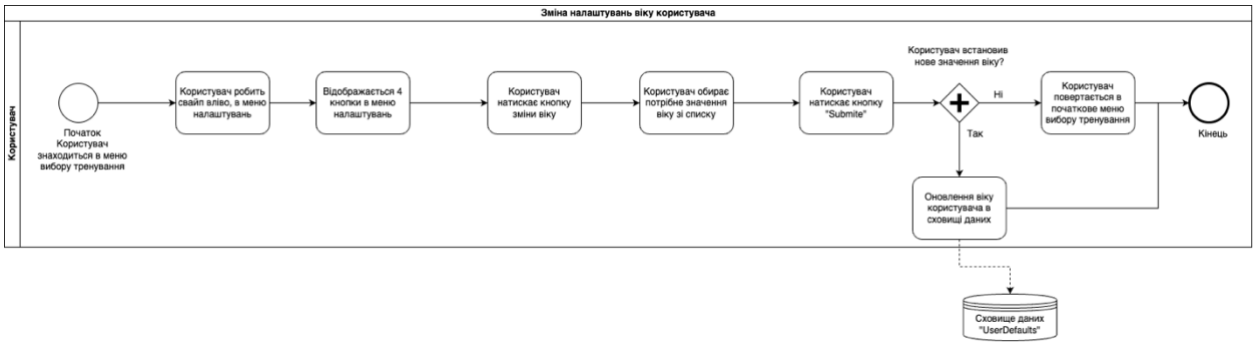


Рисунок 2.1 – Схема бізнес-процесу встановлення віку користувача

Опис послідовності встановлення віку користувача:

– користувач переходить в меню налаштувань;

– користувач натискає кнопку встановлення віку;

– користувач обирає вік зі списку наведених значень;

– Користувач натискає кнопку “Submit” і переходить в початкове меню вибору тренування;

– Якщо користувач встановив нове значення віку, це значення записується у сховище даних UserDefaults.

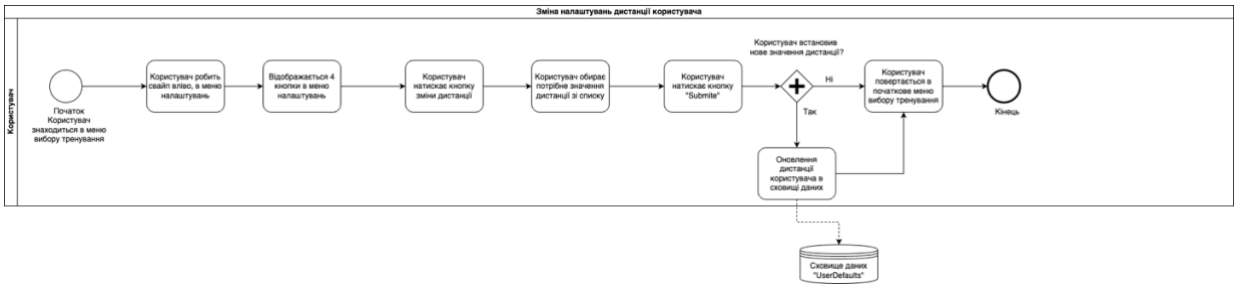


Рисунок 2.2 – Схема бізнес-процесу встановлення бажаної дистанції тренування користувача

Опис послідовності встановлення дистанції тренування користувача:

– користувач переходить в меню налаштувань;

– користувач натискає кнопку встановлення дистанції;

– користувач обирає бажану дистанцію зі списку наведених значень;

– користувач натискає кнопку “Submit” і переходить в початкове меню вибору тренування;

– якщо користувач встановив нове значення дистанції, це значення записується у сховище даних UserDefaults.

Рисунок 2.3 – Схема бізнес-процесу отримання довідки

Опис послідовності отримання довідки про застосунок:

– користувач переходить в меню налаштувань;

– користувач натискає кнопку отримання довідки;

– користувач читає довідку, натискає кнопку “Ok” і переходить в початкове меню вибору тренування.

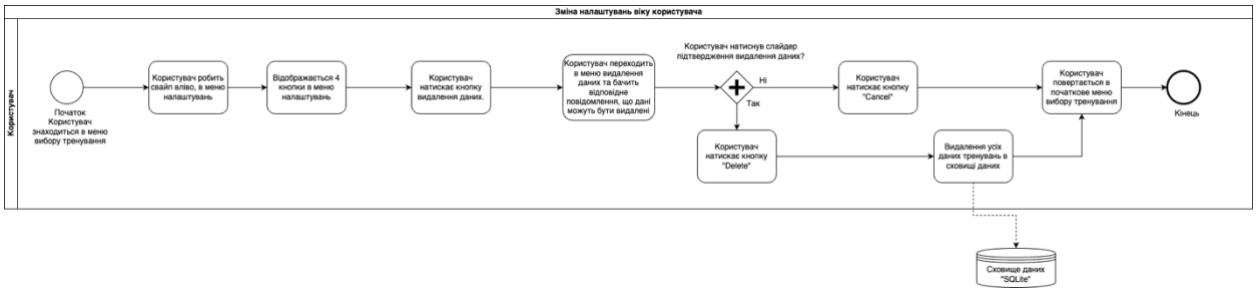


Рисунок 2.4 – Схема бізнес-процесу видалення збережених даних тренувань користувача.

Опис послідовності видалення збережених даних тренувань користувача:

– користувач переходить в меню налаштувань;

– користувач натискає кнопку видалення даних;

– користувач повинен підтвердити, що він дійсно хоче видалити дані, натиснувши відповідний слайдер;

– якщо слайдер натиснуто, користувач бачить кнопку “Delete”; при натисканні на неї дані видаляються і користувач переходить в меню вибору тренування;

– якщо слайдер не натиснуто, користувач бачить кнопку “Cancel”; при натисканні на неї користувач переходить в меню вибору тренування.

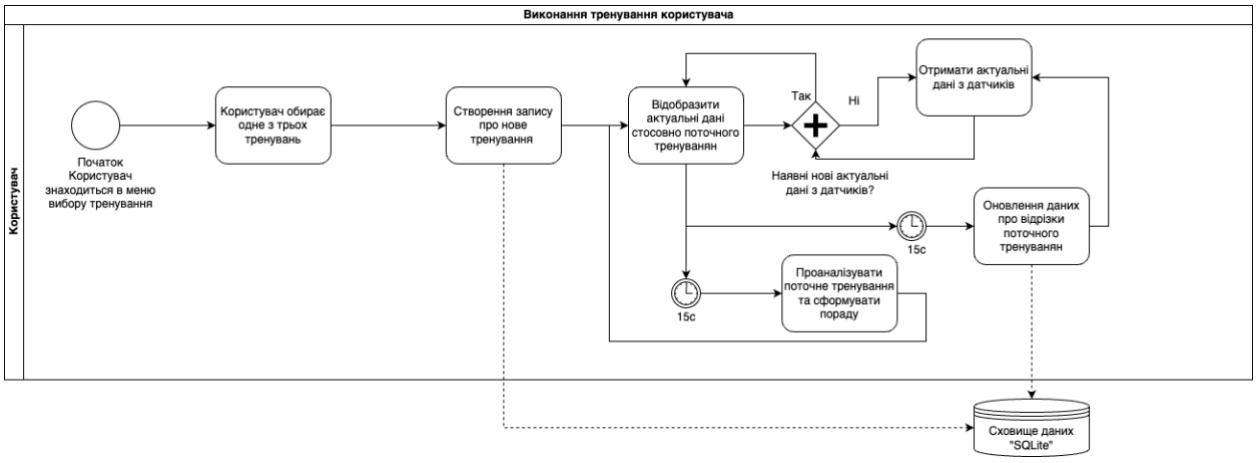


Рисунок 2.5 – Схема бізнес-процесу виконання тренування користувачем.

Опис послідовності виконання тренування користувачем:

– користувач обирає одне з трьох доступних тренувань;

– застосунок записує в базу даних дані про початок тренування;

– застосунок виводить на екран актуальні дані поточного тренування і раз на 15 секунд робить збереження цих даних в базу даних;

– застосунок раз на 15 секунд викликає алгоритм формування порад користувачу.

Рисунок 2.6 – Схема бізнес-процесу встановлення паузи користувачем

Опис послідовності встановлення паузи користувачем:

– користувач знаходиться на головному екрані активного тренування;

– користувач робить свайп вліво і потрапляє в меню опцій тренування;

– користувач натискає паузу і застосунок перестає оновлювати дистанцію, час та витрати енергії допоки користувач не натисне паузу знову.

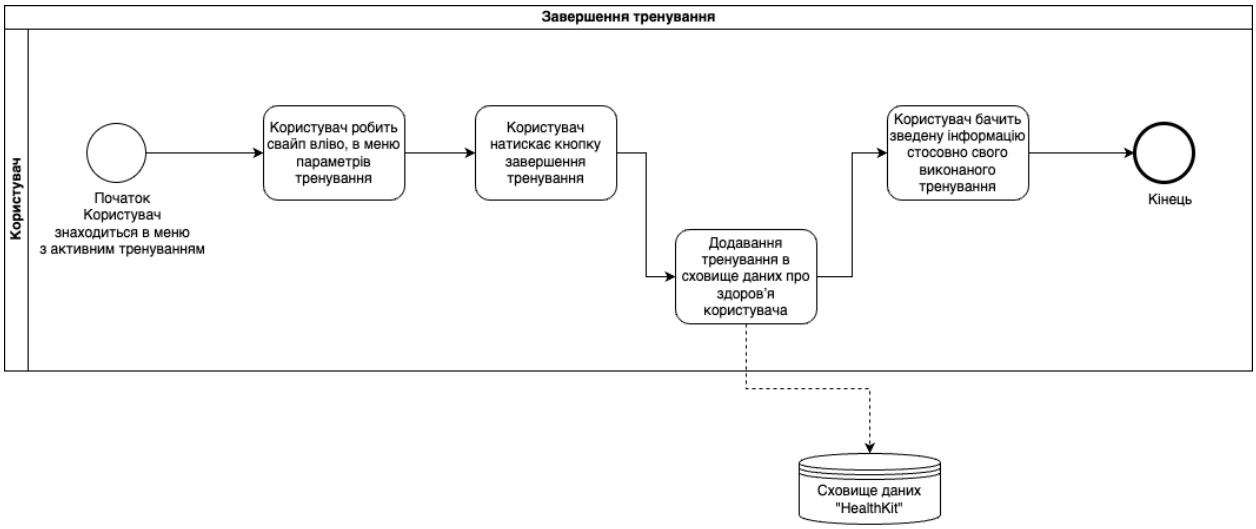


Рисунок 2.7 – Схема бізнес-процесу завершення тренування користувачем

Опис послідовності завершення тренування користувачем:

– користувач знаходиться на головному екрані активного тренування;

– користувач робить свайп вліво і потрапляє в меню опцій тренування;

– користувач натискає кнопку завершення тренування;

– застосунок зберігає дані тренування у HealthKit та виводить зведену статистику по тренуванню.

2.2 Архітектура програмного забезпечення

Сучасні мобільні застосунки часто представляють з себе великі проєкти зі складною архітектурою, розподіленою обробкою та зберіганням даних. Для полегшення розробки, а також майбутнього тестування та підтримки існують дизайн патерни. Для мобільної розробки існує три популярних патерни – MVC, MVP, MVVM. Хоча метою цього дипломного проєкту є розробка застосунку для розумного годинника на платформі WatchOS, для нього теж актуальні всі наявні переваги та недоліки трьох наведених дизайн-патернів.

MVP (Model-View-Presenter) - модель відповідає за дані та бізнес-логіку, представлення відповідає за інтерфейс користувача, а презентор виступає посередником між ними, обробляючи взаємодії користувача та оновлюючи представлення за потреби. Ключова ідея MVP полягає в тому, що View і Model повинні бути повністю відокремлені, а Presenter виступає в якості посередника. Представлення залежить тільки від даних, які встановив презентер. Перевагою цього патерну є легкість освоєння, простіша кодова база, менша кількість залежностей та сумісність зі старими технологіями. Крім переваг, патерн має і ряд недоліків, таких як більш тісний зв'язок між рівнями, складність роботи зі складними інтерфейсами користувача, більша кількість шаблонного коду, обмежене зв'язування даних і труднощі тестування. Схему патерну MVP наведено на рисунку 2.8.

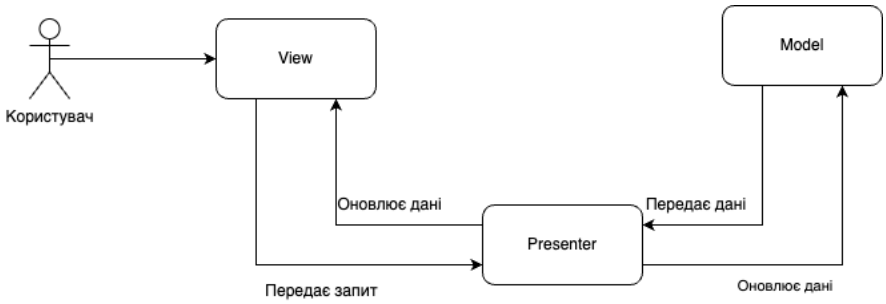


Рисунок 2.8 – Архітектура дизайн патерну MVP

MVC (Model-View-Controller) – архітектурний патерн, що розділяє програму на три основні логічні модулі: модель, представлення та контролер. Особливістю патерну є відокремлення бізнес логіки та рівня презентації один від одного. Перевагою цього патерну проєктування над MVP та MVVM є можливість паралельної розробки представлень, контролерів та моделей, що може суттєво зменшити час розробки програмного продукту при наявності команди з декількох розробників. Схему патерну MVC наведено на рисунку 2.9.

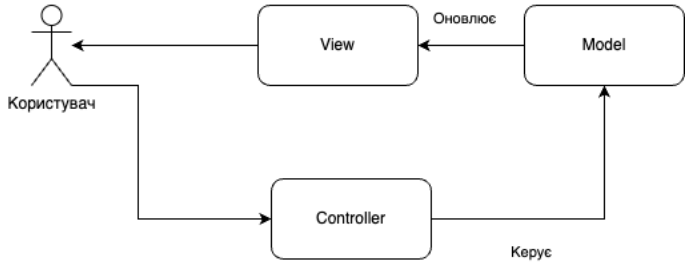


Рисунок 2.9 – Архітектура дизайн патерну MVC

MVVM (Model-View-ViewModel) – найновіший з наведених архітектурних патернів. У ньому модель(Model) відповідає за дані та бізнес-логіку, представлення(View) відповідає за користувацький інтерфейс, а представлення-модель(ViewModel) за надання даних від моделі представленню, а також за обробку введених користувачем даних та оновлення моделі. Ключова ідея MVVM полягає в тому, що ViewModel діє як посередник між View і Model, при цьому View прив'язується до властивостей ViewModel, а ViewModel оновлює модель за потреби. Незважаючи на велику схожість цього дизайн патерну з патерном MVP, існує важлива деталь, яка і робить з них два окремих дизайн-патерни. ViewModel не може напряму впливати на представлення, він виступає лише джерелом даних і через функції визову передає їх у представлення. Перевагою цього шаблону проєктування над іншими є легкість тестування та розробки окремих компонентів Завдяки тому, що ViewModel не має жодної інформації про View. View і Model також не пов'язані однин з одним, що полегшує зміну одного, не впливаючи на інший. Недоліком шаблону проєктування можуть бути проблеми з продуктивністю застосунку для великих та складних проєктів, пов'язані з методами зв'язування та передачі даних. Схему патерну MVVM представлено на рисунку 2.10.

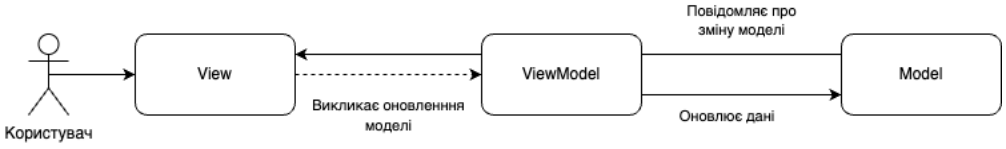


Рисунок 2.10 – Архітектура дизайн патерну MVVM

Після аналізу усіх переваг та недоліків патернів проєктування для розробки дипломного проєкту було обрано MVVM, бо переваги патернів MVC та MVP не можуть бути повною мірою застосовані до цього дипломного проєкту. В той же час, основний недолік патерну MVVM також не має суттєвого впливу на цей дипломний проєкт, бо проєкт не є великим або надвеликим [5].

Після вибору дизайну патерну, створимо на його основі схему архітектури застосунку – рисунок 2.11.

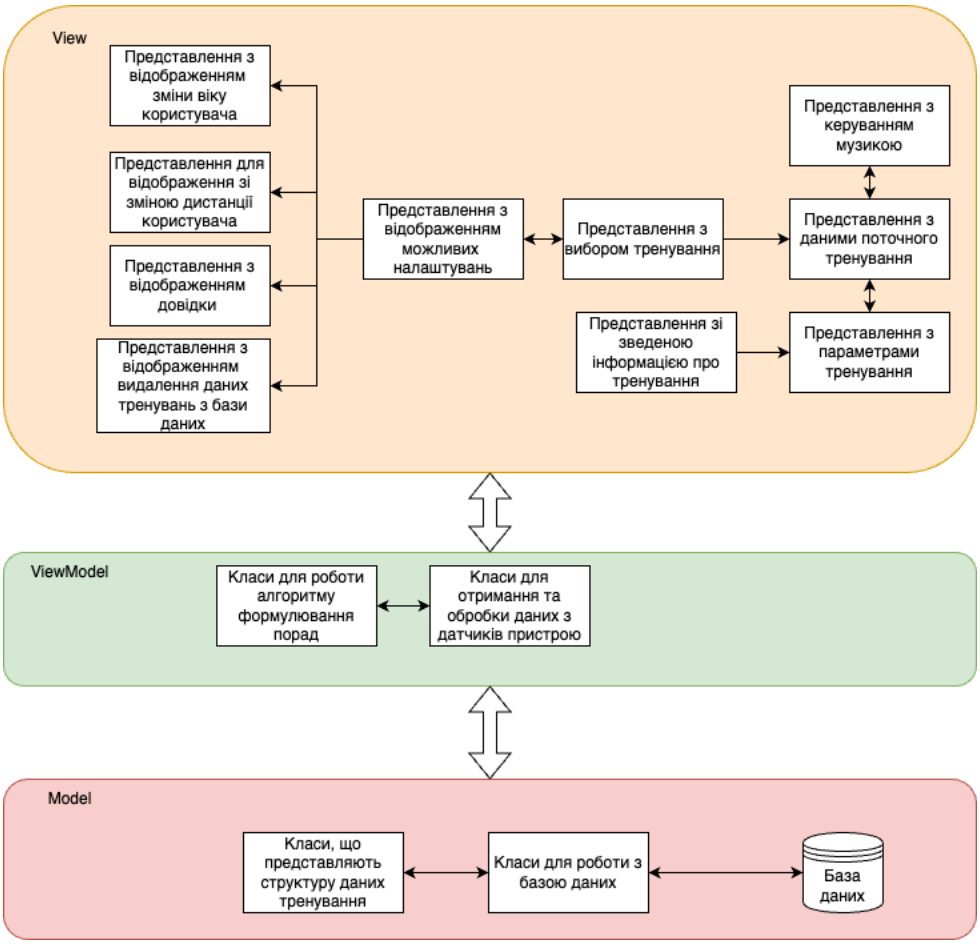


Рисунок 2.11 – Архітектура застосунку

Архітектура всього застосунку розподілена на 3 блоки: Model, ViewModel та View, як і в самому дизайн патерні MVVM. У блоці View односторонні стрілочки означають, що перехід між представленнями можливий тільки в одну сторону, через натискання кнопки. Двостороння стрілка означає, що можливий перехід між представленнями в обидва боки під час користування застосунком.

Для кращого розуміння структури проєкту перед початком розробки і написання коду слід побудувати ER діаграму сутностей. ER-діаграми є корисним інструментом у розробці проєкту, оскільки вони надають візуальне представлення сутностей, атрибутів і зав’язків, наявних у проєкті. Візуалізація цих зав’язків в свою чергу дає уяву про обсяг майбутнього проєкту, що може допомогти у виборі системи керування базами даних, архітектурного патерну чи бібліотек. ER діаграма проєкту наведена на рисунку 2.12.

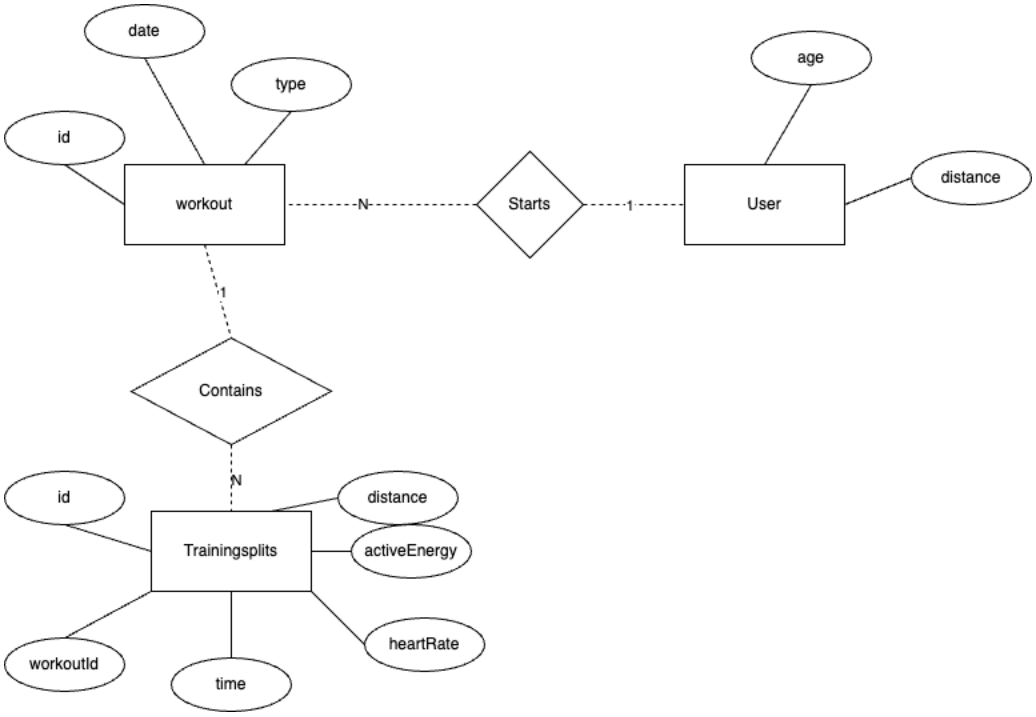


Рисунок 2.12 – ER діаграма сутностей User, Workout, Trainingsplits

Зробимо опис усіх наведених сутностей, атрибутів та зав’язків.

Сутність User представляє користувача застосунку. В системі може бути лише один користувач. Атрибути сутності User наведені в таблиці 2.1

Таблиця 2.1 – Опис атрибутів сутності User

Назва	Опис
age	Вік користувача, який потрібно вказати в налаштуваннях застосунку. Використовується для попередження про досягнення межі максимального рекомендованого пульсу.
distance	Бажана цільова дистанція для наступного тренування. Також вказується в налаштуваннях застосунку.

Сутність Workout представляє тренування користувача. Користувач може починати та виконувати багато тренувань. Атрибути сутності Workout наведені в таблиці 2.2

Таблиця 2.2 – Опис атрибутів сутності Workout

Назва	Опис
id	Ідентифікаційний номер тренування
date	Дата та час початку тренування
type	Тип тренування. Цей атрибут містить один з трьох типів тренування: «Run» - бігове тренування, «Bike» - тренування на велосипеді, «Walk» - піша прогулянка.

Сутність Trainingsplits представляє відрізки тренувань поточного тренування. Кожне тренування може містити багато відрізків тренування. Ці дані збираються та зберігаються для подальшого аналізу алгоритмом створення порад користувачу. Атрибути сутності Trainingsplits наведені в таблиці 2.3

Таблиця 2.3 – Опис атрибутів сутності Trainingsplits

Назва	Опис
id	Ідентифікаційний номер відрізку тренування.
workoutId	Ідентифікаційний номер тренування, до якого належить поточний відрізок.
time	Дата та час запису поточного відрізку тренування.
heartrate	Пульс користувача у момент, коли було записано поточний відрізок.
activeEnergy	Витрачена енергія у кілокалоріях на момент запису відрізку.
distance	Пройдена дистанція у метрах на момент запису відрізку.

2.3 Конструювання програмного забезпечення

Основним елементом, який відрізняє застосунок, розроблений в ході написання цього дипломного проєкту, від аналогів, наведених в пункті 1.3.3 є алгоритм попередження користувача про перевищення максимального рекомендованого пульсу та алгоритм формування порад користувачу стосовно продовження поточного тренування. Перший алгоритм спочатку визначає максимальний рекомендований пульс різницею числа 220 та віку користувача. Якщо поточний пульс під час тренування більше за максимальний рекомендований, на окремому екрані для порад та повідомлень користувач побачить фразу “Stop! Your heart rate is too high!”, яка буде виділена червоним кольором для привернення уваги користувача. Застереження буде відображатися стільки часу, скільки пульс користувача буде перевищувати максимальний рекомендований. Користувач має на свій розсуд вирішити, чи завершувати тренування, чи зменшити темп або зупинитися і після нормалізації пульсу продовжити тренування.

Поради користувачу відображаються на одному екрані з застереженням про високий пульс, але мають менший пріоритет. Таким чином, при перевищенні пульсу саме це застереження буде виведено на екран, а не

поточна порада. Також слід зазначити, що поради мають рекомендаційний характер і ні до чого не зобов’язують користувача

Алгоритм формування порад оснований на порівнянні даних поточного тренування з даними попередніх тренувань.

Після першого завантаження застосунку, користувач має зробити мінімум 3 тренування одного типу, перш ніж застосунок почне демонструвати поради. Також поради починають відображатися після того, як користувач подолає 25% дистанції тренування. Це необхідно для збору початкової статистики поточного тренування, на основі якої буде робитися прогнозування витрат енергії для алгоритму порівняння. Після отримання прогнозованої енергії яка буде витрачена на подолання всієї дистанції, буде виконано запит до бази даних, який має повернути усі записи з поточним типом тренування, та поточної дистанції з відхиленням 10%. Це необхідно для фіксування тільки значущих відхилень у витратах енергії, адже при відсутності подібного діапазону, алгоритм постійно б змінював поради при проведенні однакових тренувань, що б суттєво відволікало користувача. Після цього алгоритм порівнює прогнозовану енергію з отриманим діапазоном і рекомендує зменшити темп, якщо прогнозовані витрати енергії перевищують верхню межу діапазону, рухатись в такому ж темпі, якщо прогнозоване значення належить діапазону і збільшити темп, якщо прогнозоване значення менше нижньої межі діапазону.

Окремо слід описати ситуацію, коли користувач ставить за мету подолати дистанцію, яка перевищує максимальну дистанцію раніше записаного тренування. В такому разі алгоритм побудує прогноз витрат енергії для максимальної раніше записаної дистанції і додатково ще раз врахує 10% відхилення від витрат енергії, але буде показувати тільки 2 можливі поради: «Продовжувати з тим самим темпом», або «Трохи знизити темп». Таке рішення було обрано, бо за відсутності інформації про можливу максимальну дистанцію, радити користувачу «збільшити темп» для досягнення нового рекорду може бути шкідливим.

Такий алгоритм було розроблено, бо застосунок орієнтований на людей, які тільки починають займатися циклічними видами спорту (біг, велосипед, прогулянка) і хочуть поступово збільшувати дистанцію своїх тренувань, не переймаючись за темп проведення цих тренувань. За таких обставин успіх у подоланні запланованої дистанції дуже залежить від першої чверті тренування, бо якщо не тренований користувач почне виконувати тренування з занадто високим темпом, це одразу призведе до різкого підвищення пульсу, що в комплексі буде означати більші витрати енергії і меншу вірогідність подолати заплановану дистанцію. Блок-схема алгоритму формування порад наведена на рисунку 2.13.

Раніше описаний алгоритм формування порад оснований на порівнянні даних раніше проведених тренувань з поточними даними. Дані представляють чітку структуру, тому в якості системи управління базами даних буде використана SQLite. До переваг цієї бази даних можна віднести невеликий розмір швидкодію і високу гнучкість. SQLite — це невеликий і швидкий механізм баз даних, який може ефективно обробляти великі обсяги даних, що робить його ідеальним вибором для додатків для розумних годинників, які мають обмежені можливості зберігання та обробки даних. Також великою перевагою є легкість інтегрування бази даних в проєкт та легкість використання.

Також слід одразу визначити недоліки обраної бази даних. SQLite це однопоточна база даних, що означає неможливість виконання двох і більше операцій одночасно. При наявності декількох користувачів це могло б бути проблемо, однак цей застосунок спроєктований таким чином, що немає необхідності виконувати будь-які операції одночасно. Ще одним недоліком є проблеми з масштабованістю бази даних. Вони можуть виникнути, коли застосунок швидко розробляється і його функціонал сильно збільшується в процесі розробки. Це також не є актуальною проблемою для цього дипломного проєкту, адже усі таблиці та зв’язки бази даних визначаються на етапі проєктування і не будуть змінюватися з часом.

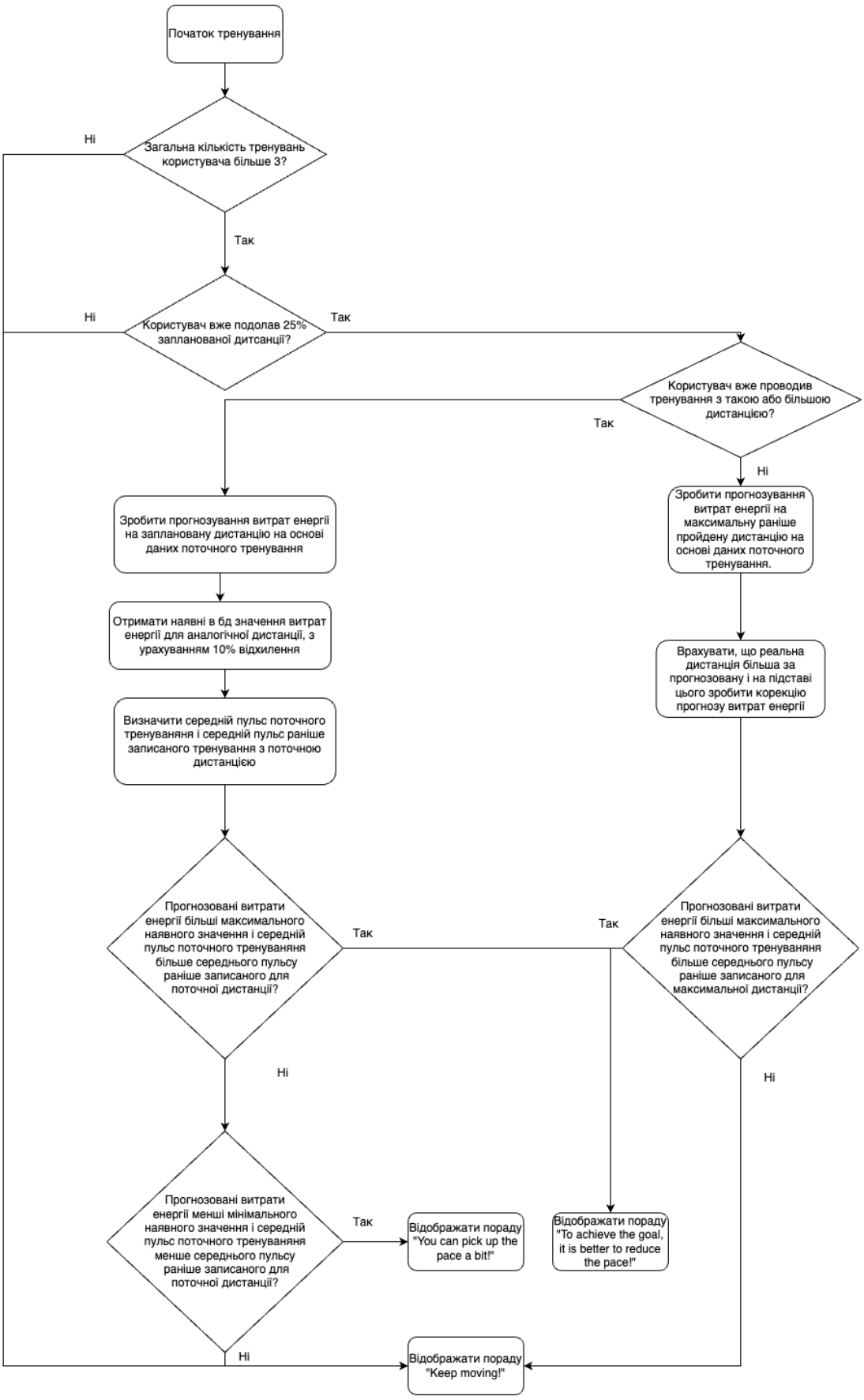


Рисунок 2.13 – Блок-схема алгоритму формування порад користувачу

Опис таблиць бази даних наведено у таблицях 2.4 - 2.5. Модель бази даних наведена на рисунку 2.12

Таблиця 2.4 – Опис таблиці workout

Таблиця	Назва поля	Тип даних	Опис
workout	id	INTEGER	ідентифікаційний номер тренування
	date	TEXT	Дата та час початку тренування
	type	TEXT	Тип тренування

Таблиця 2.5 – Опис таблиці trainingsplits

Таблиця	Назва поля	Тип даних	Опис
trainingsplits	id	INTEGER	ідентифікаційний номер відрізку тренування
	workoutId	TEXT	ідентифікаційний номер тренування, до якого відноситься поточний відрізок
	time	TEXT	Тип тренування
	heartRate	DOUBLE	Поточний пульс спортсмена
	activeEnergy	DOUBLE	Витрачена енергія у кілокалоріях
	distance	DOUBLE	Пройдена дистанція

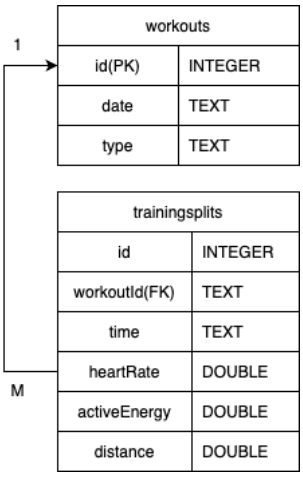


Рисунок 2.12 – Модель бази даних

Модель бази даних відрізняється від ER діаграми на рисунку 2.2 відсутністю таблиці user.

Сутність user має тільки 2 атрибути – age та distance, які відображають вік користувача та бажану дистанцію тренування. Для зберігання цих даних буде використовуватися механізм параметрів користувача. Завдяки цьому механізму можна зберігати невеликі обсяги даних у фізичній пам’яті девайсу.

Стандартні параметри користувача представлені класом UserDefaults у Swift, який забезпечує простий інтерфейс для читання та запису даних у систему. Система за замовчуванням використовує серіалізацію списку властивостей, що означає, що дані можуть зберігатися в різних форматах, включаючи рядки, числа, масиви та словники.

Опис основних класів та інтерфейсів застосунку наведений у таблиці 2.6

Таблиця 2.6 – Основні класи та інтерфейси застосунку

Назва Інтерфейсу	Опис
StartView	Стартове меню застосунку, яке вказує на інтерфейс, який першим відображається після збірки застосунку.
SettingsView	Відображає меню налаштувань застосунку. Містить кнопки вибору віку, дистанції, інформації, видалення даних бази даних

Продовження таблиці 2.6

AgeRequestView	Відображає сторінку вибору віку користувача
DistanceRequestView	Відображає сторінку вибору цільової дистанції наступного тренування
InfoView	Відображає сторінку з інформацією про застосунок
DeleteDataView	Відображає сторінку з меню видалення даних тренувань з бази даних застосунка
MainView	Відображає меню з основними метриками тренування та порадами користувачу стосовно поточного тренування
SessionPagingView	Відображає структуру, в якому порядку мають відображатися вікна застосунку під час тренування.
TrainingViewModel	Клас – який відповідає за всі обчислення та дані, пов’язані з тренуванням
LinearRegression	Клас – який відповідає за лінійну регресію
SqLiteDb	Клас – який відповідає за доступ до бази даних і обмін інформацією з нею.
WorkoutModel	Клас – який визначає структуру тренування
TrainingSplits	Клас – який визначає структуру відрізків тренування
EnumConstants	Клас – який містить усі текстові повідомлення, які можуть відображатися в застосунку

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.7

Таблиця 2.7 – Опис утиліт та бібліотек

№ п/п	Назва утиліти	Опис застосування
1	XCode	Головне середовище розробки програмного забезпечення серверної частини курсової роботи.
2	WatchOs emulator	Програмне забезпечення для емулювання розумного годинника. Дозволяє легко та швидко тестувати програмне забезпечення без необхідності кожен раз встановлювати його на фізичний девайс, що дуже сильно економить час розробки та тестування. Також важливою особливістю є можливість емулювати будь-який з підтримуваних годинників, що дозволяє тестувати адаптивність користувацького інтерфейсу під різні діагоналі.
3	SwiftUI	SwiftUI надає усі необхідні елементи для побудови користувацького інтерфейсу застосунку. Фреймворк дозволяє обробляти події, пов’язані з натисканням кнопок, а також керувати потоками даних в застосунку.
4	HealthKit	Фреймворк HealthKit використовується для роботи з даними, пов’язаними зі здоров’ям користувача. Фреймворк обмежує типи даних і одиниць попередньо визначеним списком, гарантуючи, що всі програми розуміють значення даних і як їх можна використовувати.

2.4 Аналіз безпеки даних

Безпека даних – тема, яка повинна бути вивчена та проаналізована при побудові будь-якого застосунку. Головною ціллю захисту даних є убезпечення

чутливих даних, таких як логіни, паролі, ідентифікаційні, або всіх інших, що можуть завдати шкоди користувачу при потраплянні до зловмисників.

У застосунку цього дипломного проєкту існує 3 типи сховищ даних:

– UserDefaults – зберігає вік та дистанцію користувача;

– SQLite – база даних, яка зберігає дані тренувань користувача;

– HealthKit – бібліотека, яка централізовано зберігає всі дані, які відносяться до здоров'я користувача і може надавати їх стороннім застосункам.

Серед трьох наведених типів даних тільки HealthKit може містити подібні дані, які користувач міг занести туди через інший застосунок. Про безпеку цих даних подбала сама компанія Apple.

HealthKit вимагає дозволу користувача, щоб будь-яка програма могла отримати доступ до даних про здоров'я. Користувачі можуть вибирати, якими даними ділитися з кожною програмою, і можуть будь-коли скасувати доступ. Таким чином, не треба надавати доступ підозрілим програмам до чутливих даних.

Крім безпеки, пов'язаної з наданням доступу до даних тільки перевіреним програмам, усі дані, що зберігаються в HealthKit, зашифровані як під час передачі, так і під час зберігання. HealthKit використовує стандартні протоколи шифрування для захисту даних користувачів. Додатково до вищеописаного слід відмітити, що HealthKit зберігає дані про здоров'я окремо від даних інших програм. Це означає, що навіть якщо програму зламано, дані про здоров'я залишаються ізольованими та захищеними.

Загалом компанія Apple дуже серйозно ставиться до захисту даних і запровадила низку заходів безпеки, щоб забезпечити безпеку та конфіденційність даних користувачів у бібліотеці HealthKit та своїй операційній системі в цілому, бо потенційна вірусна програма спочатку має бути встановлена на пристрій користувача, що можливо зробити тільки через

офіційний магазин застосунків Watch App Store, який в свою чергу теж перевіряє усі програми на наявність вірусів.

Висновки до розділу

У другому розділі дипломного проєкту було виконано моделювання та конструювання програмного забезпечення.

Описані всі бізнес процеси та проілюстровані завдяки BPMN діаграмам. Такі дії допомагають краще розуміти систему в цілому, що дуже пришвидшить в майбутньому процес розробки та написання коду.

Також у цьому розділі було порівняно три найбільш популярні дизайн патерни для мобільної розробки. Розробка мобільних застосунків дуже схожа на розробку застосунків для розумних годинників, завдяки чому застосування одного з трьох патернів (MVVM, MVC або MVP) є доречним. Після аналізу і порівняння всіх переваг та недоліків було обрано патерн MVVM, адже в поточному проєкті є можливості скористатися перевагами такої структури, при цьому існуючі недоліки не актуальні у випадку цього проєкту.

Одним з важливих елементів цього застосунку для розумних годинників для аналізу фізичної активності є алгоритм формування порад, основною ідеєю якого є прогнозування витрат енергії для поточного тренування і порівняння цього прогнозованого значення з вже наявними значеннями, записаними протягом попередніх тренувань, які застосунок зберігає у спеціально спроєктовану базу даних SQLite.

Наприкінці розділу було розглянуто безпеку даних. До чутливих даних користувача можливо потрапити тільки через бібліотеку HealthKit, яка в свою чергу дуже добре захищена від зламів та витоку даних компанією Apple.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості коду — це процес оцінки вихідного коду на потенційні дефекти, дотримання стандартів та потенційні проблеми з безпекою. Цей аналіз зазвичай виконується за допомогою спеціалізованих програмних інструментів, призначених для аналізу коду та надання звіту стосовно його якості.

Інструменти аналізу якості коду вимірюють різні показники для оцінки якості коду. Ці показники можуть включати рядки коду, дублювання коду, покриття коду, запахи коду. Аналізуючи ці показники, розробники та команди можуть отримати уявлення про зручність обслуговування, читабельність, ефективність і потенційні проблеми їх коду.

Інструменти аналізу якості коду часто включають перевірки, орієнтовані на безпеку, щоб виявити потенційні вразливості в коді. Ці інструменти можуть виявляти такі поширені недоліки безпеки, як запис статичних логінів та паролей, незахищені методи шифрування та незахищена конфігурація. Усунувши ці вразливості на ранніх стадіях розробки, розробники можуть підвищити рівень безпеки своїх програм.

Інструменти аналізу якості коду можна інтегрувати із середовищами розробки, системами безперервної інтеграції і системами контролю версій. Це дозволяє розробникам автоматизувати процес аналізу коду та інтегрувати його у процес розробки.

Інструменти аналізу якості коду надають звіти, інформаційні панелі та візуалізації для представлення результатів аналізу. Ці звіти висвітлюють виявлені проблеми, показники, тенденції та містять рекомендації щодо покращення. Така інформація допомагає розробникам і командам визначати пріоритети в своїй роботі, відстежувати прогрес і приймати обґрунтовані рішення щодо подальшої розробки та обслуговування.

Загалом, аналіз якості коду за допомогою спеціального програмного забезпечення є цінною практикою в сучасній розробці програмного забезпечення. Це допомагає командам переконатися, що їх кодова база є чистою, зручною для обслуговування, безпечною та відповідає галузевим стандартам. Використовуючи інструменти аналізу якості коду, розробники можуть завчасно виявляти потенційні проблеми, покращувати загальну якість свого коду та створювати більш надійне програмне забезпечення [6].

Для аналізу якості коду цього дипломного проєкту буде використаний сервіс Embold. Сервіс має зручну інтеграцію з найпопулярнішими сервісами контролю версій, що дозволяє виконувати аналіз програмного коду проєктку безпосередньо у репозиторії, без необхідності встановлення програмного забезпечення локально.

На основі проаналізованого коду програма генерує звіт – рисунок 3.1.

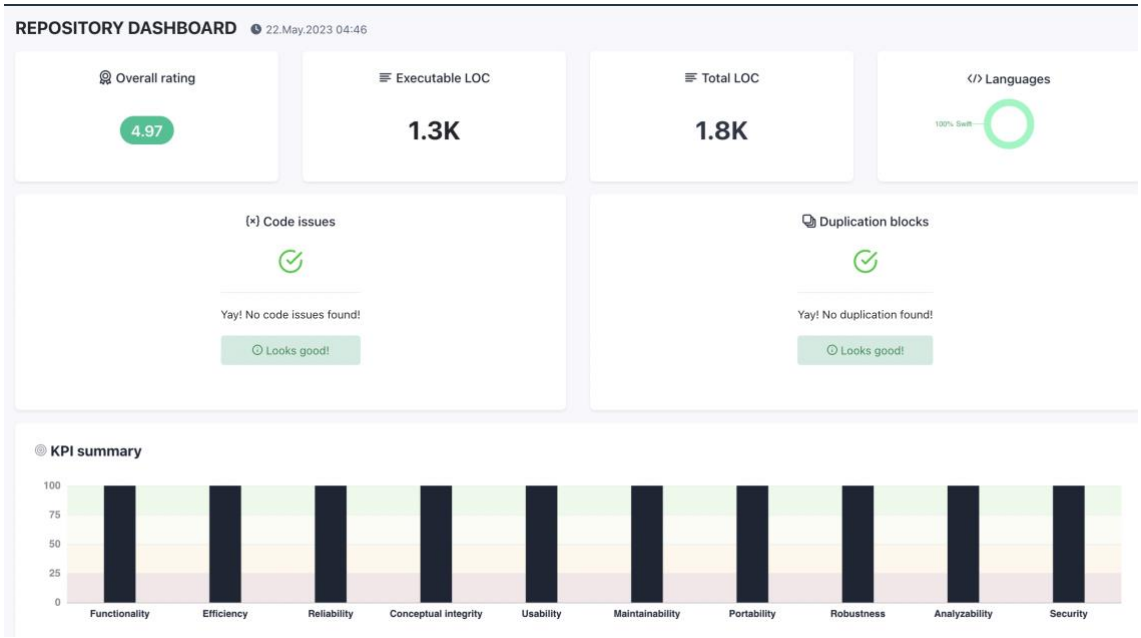


Рисунок 3.1 – Звіт про проаналізований код

Проєкт було проаналізовано за критеріями:

– Functionality (Функціональність)
– Efficiency (Ефективність)
– Reliability (Надійність)
– Conceptual integrity (Концептуальна цілісність)
– Usability (Зручність використання)

– Maintainability (Легкість підтримки)
– Portability (Портативність)
– Robustness (Стійкість до відмов)
– Analyzability (Придатність до аналізу)
– Security (Безпека)

Кожен з цих критеріїв був оцінений аналізатором на 100 балів зі 100 можливих.

Також аналізатор коду аналізує кожний наявний файл на якість коду, що включає в себе аналіз безпеки, аналіз дуплікацій коду, наявність багів та потенційних проблем. Середня оцінка усіх файлів формує загальний рейтинг і для даного проєкту склала 4.97 одиниць з 5 можливих.

Загалом, базуючись на звіті аналізатору коду Embold, код дипломного проєкту був оцінений як дуже хороший, без будь-яких багів, дуплікацій та проблем з безпекою.

3.2 Опис процесів тестування

Мануальне тестування — це підхід до тестування програмного забезпечення, коли тестувальники виконують тестові сценарії вручну, без використання автоматизованих інструментів чи сценаріїв. Такий підхід передбачає задіяння людини для перевірки функціональності програмного забезпечення, виявлення дефектів і оцінки загальної якості системи. Мануальне тестування є важливою частиною життєвого циклу розробки програмного забезпечення (SDLC) і зазвичай виконується спеціальною людиною – мануальним тестувальником [7]. Визначимо основні переваги мануального тестування.

Мануальне тестування особливо ефективне для дослідницького тестування, коли тестувальники активно досліджують програмне забезпечення, щоб знайти помилки, проблеми зі зручністю використання та інші потенційні проблеми. Вони можуть мислити творчо, імітувати поведінку

користувачів і виявляти несподівані проблеми, які не були передбачені на етапі розробки.

Мануальне тестування дає змогу тестувальникам оцінювати досвід користувача безпосередньо взаємодіючи з застосунком, як це робили б кінцеві користувачі. Вони можуть оцінювати такі фактори, як інтуїтивність, простота використання, швидкість реагування та загальне задоволення користувачів. Цей відгук є цінним для покращення зручності використання програмного забезпечення та дизайну інтерфейсу користувача.

Мануальне тестування ефективне для оцінки нефункціональних аспектів програмного забезпечення, таких як продуктивність, безпека, сумісність і локалізація. Тестувальники можуть відтворювати реальні сценарії роботи застосунку та спостерігати за поведінкою системи.

Недоліком мануального тестування є необхідність проведення повторних тестів кожен раз, коли програмне забезпечення зазнало змін. У випадку даного дипломного проєкту тестування виконується після остаточного завершення розробки, що нівелює цей недолік.

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.13.

Таблиця 3.1 – Тест 1.1

Тест	Встановлення віку користувача
Модуль	Налаштування
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці налаштувань
Вхідні дані	Вік користувача
Опис проведення тесту	Користувач натискає кнопку зміни віку та переходить в меню зміни віку користувача. Обирає зі списку варіантів потрібне значення віку та натискає кнопку «Submit».

Продовження таблиці 3.1

Очікуваний результат	Користувач повертається на головну сторінку налаштувань. Вік користувача в системі успішно змінено
Фактичний результат	Користувач повертається на головну сторінку налаштувань. Вік користувача в системі успішно змінено

Таблиця 3.2 – Тест 2.1

Тест	Встановлення запланованої дистанції тренування
Модуль	Налаштування
Номер тесту	2.1
Початковий стан системи	Користувач знаходиться на сторінці налаштувань
Вхідні дані	Дистанція тренування
Опис проведення тесту	Користувач натискає кнопку зміни дистанції та переходить в меню зміни запланованої дистанції користувача. Обирає зі списку варіантів потрібне значення дистанції та натискає кнопку «Submit».
Очікуваний результат	Користувач повертається на головну сторінку налаштувань. Запланована дистанція користувача в системі успішно змінена.
Фактичний результат	Користувач повертається на головну сторінку налаштувань. Запланована дистанція користувача в системі успішно змінена.

Таблиця 3.3 – Тест 3.1

Тест	Отримання довідки про застосунок
Модуль	Налаштування
Номер тесту	3.1

Продовження таблиці 3.3

Початковий стан системи	Користувач знаходиться на сторінці налаштувань
Вхідні дані	Відсутні
Опис проведення тесту	Користувач натискає кнопку отримання інформації про застосунок та переходить у меню, де відображається довідка. Після ознайомлення з довідкою користувач натискає кнопку «Ok»
Очікуваний результат	Користувач повертається на головну сторінку налаштувань. Інформація про застосунок успішно відображена
Фактичний результат	Користувач повертається на головну сторінку налаштувань. Інформація про застосунок успішно відображена

Таблиця 3.4 – Тест 4.1

Тест	Видалення даних тренувань з бази даних
Модуль	Налаштування
Номер тесту	4.1
Початковий стан системи	Користувач знаходиться на сторінці налаштувань
Вхідні дані	Відсутні
Опис проведення тесту	Користувач натискає кнопку видалення даних та переходить в меню для видалення даних. Після цього натискає кнопку “Cancel”
Очікуваний результат	Користувач повертається на головну сторінку налаштувань. Дані в системі не видалено.
Фактичний результат	Користувач повертається на головну сторінку налаштувань. Дані в системі не видалено.

Таблиця 3.5 – Тест 4.2

Тест	Видалення даних тренувань з бази даних
Модуль	Налаштування
Номер тесту	4.2
Початковий стан системи	Користувач знаходиться на сторінці налаштувань
Вхідні дані	Відсутні
Опис проведення тесту	Користувач натискає кнопку видалення даних та переходить в меню для видалення даних. Користувач натискає слайдер з підтвердженням видалення даних. Після цього натискає кнопку “Delete”
Очікуваний результат	Користувач повертається на головну сторінку налаштувань. Дані в системі успішно видалено.
Фактичний результат	Користувач повертається на головну сторінку налаштувань. Дані в системі успішно видалено.

Таблиця 3.6 – Тест 5.1

Тест	Запуск тренування
Модуль	Стартове меню
Номер тесту	5.1
Початковий стан системи	Користувач знаходиться на сторінці вибору тренування
Вхідні дані	Необхідне тренування
Опис проведення тесту	Користувач натискає на кнопку з необхідним йому тренуванням.
Очікуваний результат	Обране тренування починається. Користувач починає бачити як змінюються значення параметрів на головному екрані тренування

Продовження таблиці 3.6

Фактичний результат	Обране тренування починається. Користувач починає бачити як змінюються значення параметрів на головному екрані тренування

Таблиця 3.7 – Тест 6.1

Тест	Встановлення паузи на поточне тренування
Модуль	Вікно керування тренуванням
Номер тесту	6.1
Початковий стан системи	Користувач знаходиться на сторінці керування тренуванням
Вхідні дані	Відсутні
Опис проведення тесту	Користувач натискає на кнопку паузи
Очікуваний результат	Іконка на кнопці паузи змінюється. В меню з параметрами тренування зупиняється відлік часу, дистанції та калорій.
Фактичний результат	Іконка на кнопці паузи змінюється. В меню з параметрами тренування зупиняється відлік часу, дистанції та калорій.

Таблиця 3.8 – Тест 6.2

Тест	Зняття паузи на поточне тренування
Модуль	Вікно керування тренуванням
Номер тесту	6.2
Початковий стан системи	Користувач знаходиться на сторінці керування тренуванням. Активна пауза.
Вхідні дані	Відсутні

Продовження таблиці 3.8

Опис проведення тесту	Користувач натискає на кнопку паузи.
Очікуваний результат	Іконка на кнопці паузи змінюється. В меню з параметрами тренування продовжується відлік часу, дистанції та калорій.
Фактичний результат	Іконка на кнопці паузи змінюється. В меню з параметрами тренування продовжується відлік часу, дистанції та калорій.

Таблиця 3.9 – Тест 7.1

Тест	Завершення тренування
Модуль	Вікно керування тренуванням
Номер тесту	7.1
Початковий стан системи	Користувач знаходиться на сторінці керування тренуванням.
Вхідні дані	Відсутні
Опис проведення тесту	Користувач натискає на кнопку завершення тренування
Очікуваний результат	Тренування завершується. Відображається екран зі зведеною статистикою пройденого тренування.
Фактичний результат	Тренування завершується. Відображається екран зі зведеною статистикою пройденого тренування.

Таблиця 3.10 – Тест 8.1

Тест	Не надання доступу до даних Healthkit
Модуль	Перший запуск застосунку

Продовження таблиці 3.10

Номер тесту	8.1
Початковий стан системи	Користувач завантажив застосунок і перший раз відкриває його.
Вхідні дані	Відсутні
Опис проведення тесту	Користувач відкриває застосунок і бачить запит на запис та читання даних в HealthKit. Користувач не надає доступ до даних і починає тренування.
Очікуваний результат	Дані з датчиків не відображаються, при спробі завершити тренування, тренування не може бути збережено та додано до списку проведених тренувань
Фактичний результат	Дані з датчиків не відображаються, при спробі завершити тренування, тренування не може бути збережено та додано до списку проведених тренувань

Таблиця 3.11 – Тест 8.2

Тест	Надання доступу до даних Healthkit
Модуль	Перший запуск застосунку
Номер тесту	8.2
Початковий стан системи	Користувач завантажив застосунок і перший раз відкриває його.
Вхідні дані	Відсутні
Опис проведення тесту	Користувач відкриває застосунок і бачить запит на запис та читання даних в HealthKit. Користувач надає доступ до даних і починає тренування.
Очікуваний результат	Дані з датчиків відображаються, при спробі завершити тренування, тренування успішно завершується і користувач бачить зведену інформацію стосовно проведеного тренування.

Продовження таблиці 3.11

Фактичний результат	Дані з датчиків відображаються, при спробі завершити тренування, тренування успішно завершується і користувач бачить зведену інформацію стосовно проведеного тренування.

Для проведення наступного блоку мануальних тестувань алгоритму аналізу фізичної активності було проведено декілька тестових забігів з наступними параметрами:

– один забіг на дистанцію 1 км за час 5 хв з середнім пульсом 190 ударів на хвилину;

– другий забіг на дистанцію 2 км за час 12 хв з середнім пульсом 185 ударів на хвилину;

– третій забіг на дистанцію 3 км за час 21 хв з середнім пульсом 180 ударів на хвилину.

Таким чином, було додано у БД мінімально необхідну кількість даних для роботи алгоритму формування порад користувачу.

Тепер виконаємо мануальне тестування алгоритму, з ціллю отримати пораду знизити темп або підвищити темп для досягнення запланованої дистанції тренування – таблиці 3.12 - 3.13.

Таблиця 3.12 – Тест 9.1

Тест	Отримання поради про зниження темпу тренування
Модуль	Екран порад та застережень
Номер тесту	9.1
Початковий стан системи	Користувач встановив ціль тренування як 3000м, почав тренування та знаходиться на екрані порад та застережень.
Вхідні дані	Користувач подолав перші 750м за 3 хвилини 30 секунд.

Продовження таблиці 3.12

Опис проведення тесту	Користувач встановив ціль пробігти 3000м, але пробіг перші 750м набагато швидше, ніж попередній раз, коли він успішно подолав 3000м
Очікуваний результат	Застосунок відобразив рекомендацію зменшити темп.
Фактичний результат	Застосунок відобразив рекомендацію зменшити темп.

Таблиця 3.13 – Тест 9.2

Тест	Отримання поради про підвищення темпу тренування
Модуль	Екран порад та застережень
Номер тесту	9.2
Початковий стан системи	Користувач встановив ціль тренування як 1000м, почав тренування та знаходиться на екрані порад та застережень.
Вхідні дані	Користувач подолав перші 250м за 2 хвилини.
Опис проведення тесту	Користувач встановив ціль пробігти 1000м, але пробіг перші 250м набагато повільніше, ніж попередній раз, коли він успішно подолав 1000м
Очікуваний результат	Застосунок відобразив рекомендацію підвищити темп.
Фактичний результат	Застосунок відобразив рекомендацію підвищити темп.

Таким чином, алгоритм аналізу тренування та формування порад розуміє, коли користувач витрачає занадто багато енергії для досягнення мети, або рухається повільно і може трішки збільшити темп.

3.3 Опис контрольного прикладу

Опишемо максимально детальний контрольний приклад, у якому відобразимо всі можливості та особливості застосунку.

Після першого запуску застосунку користувач побачить повідомлення про необхідність надати застосунку доступ до даних здоров'я Health – рисунок 3.2

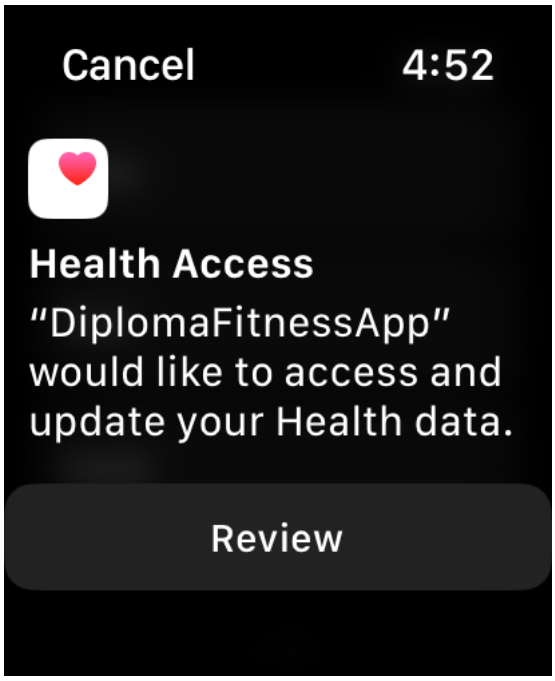


Рисунок 3.2 – Запит застосунку на доступ до даних

Після натискання кнопки "Review", у користувача з'явиться можливість надати окремо дозвіл на читання даних з застосунку Health та на запис даних – рисунки 3.3 - 3.4.

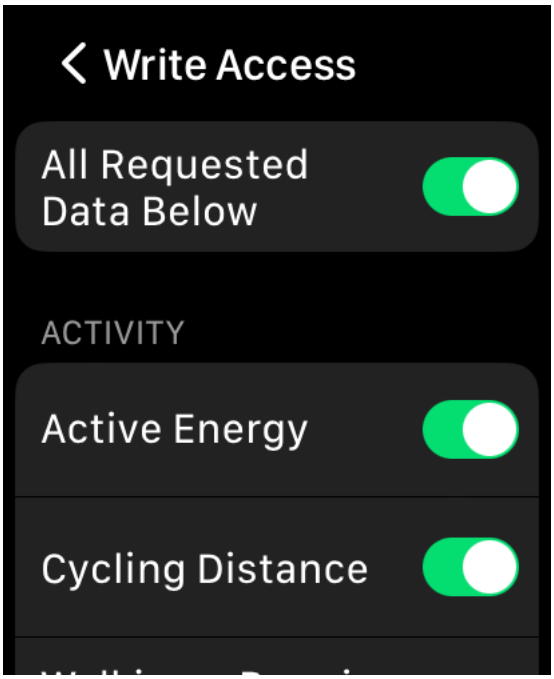


Рисунок 3.3 – Надання доступу на запис даних.

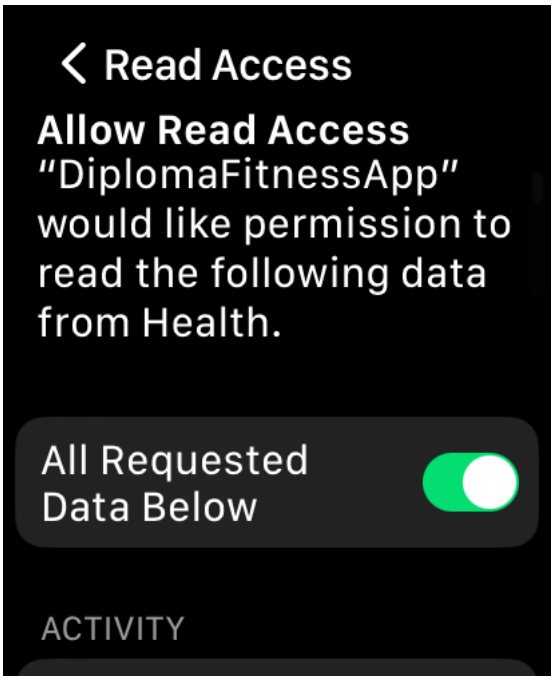


Рисунок 3.4 – Надання доступу на читання даних

Для коректної роботи застосунку необхідно надати доступ до всіх даних, про які запитує застосунок.

Після надання даних користувач потрапляє на головну сторінку вибору тренування – рисунок 3.5.

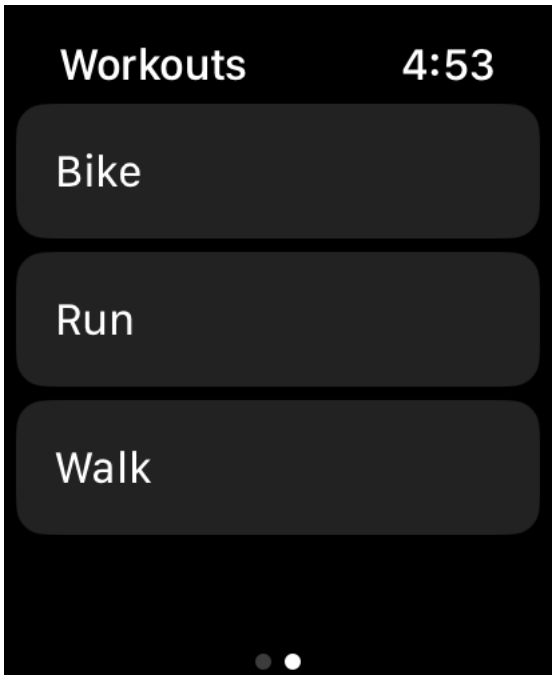


Рисунок 3.5 – Головне меню з наявними тренуваннями

Через те, що це перша взаємодія користувача з застосунком, перед початком тренування необхідно встановити налаштування користувача. Це можна зробити, перейшовши в меню налаштувань лівим свайпом – рисунок 3.6.

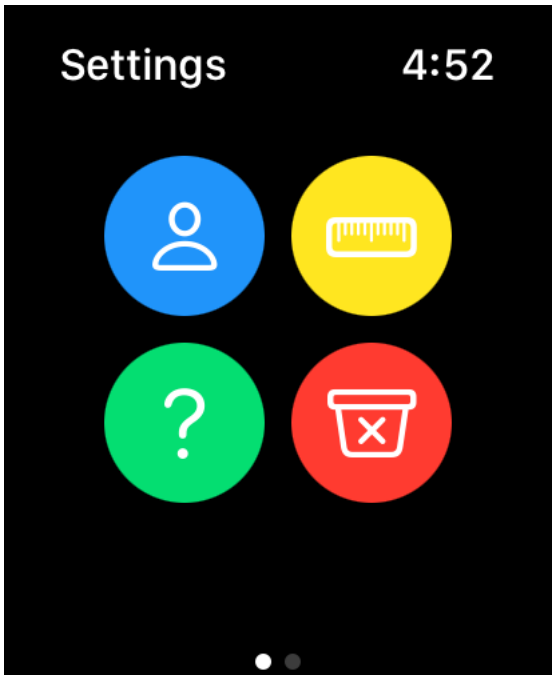


Рисунок 3.6 – Меню налаштувань застосунку

В меню доступні 4 кнопки:

– синя з іконкою людини – налаштування віку користувача;

– жовта з іконкою людини – налаштування віку користувача;

– зелена з іконкою питального знаки – отримання довідки про застосунок;

– червона з іконкою корзини – видалення всіх даних тренувань з бази даних.

Натиснувши синю кнопку відкриємо меню налаштування віку користувача – рисунок 3.7.

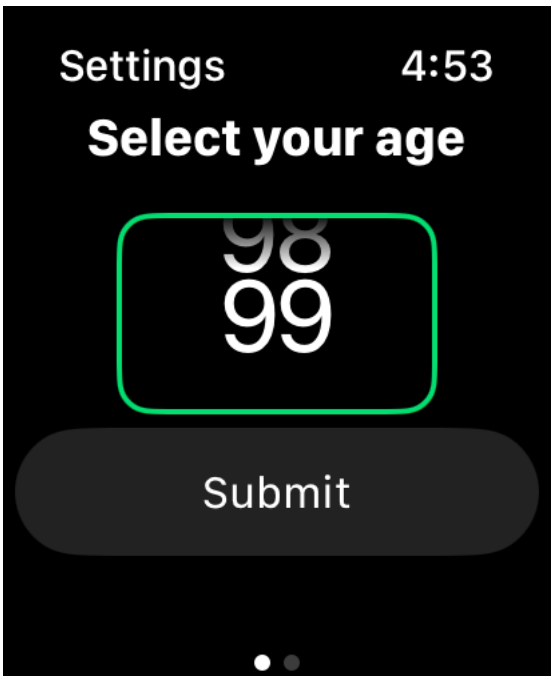


Рисунок 3.7 – Меню налаштування віку користувача

Після вибору потрібного параметру натискаємо кнопку «Submit», вік користувача зберігається в застосунку і користувач повертається до екрану налаштувань.

Аналогічно встановимо дистанцію тренування користувача – рисунок 3.8.

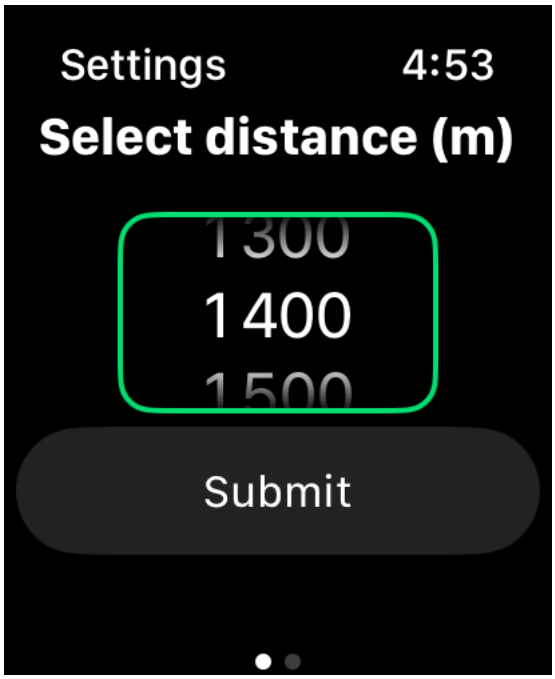


Рисунок 3.7 – Меню налаштування дистанції користувача

В меню отримання довідки користувач може прочитати коротку інформацію про застосунок – рисунок 3.8.

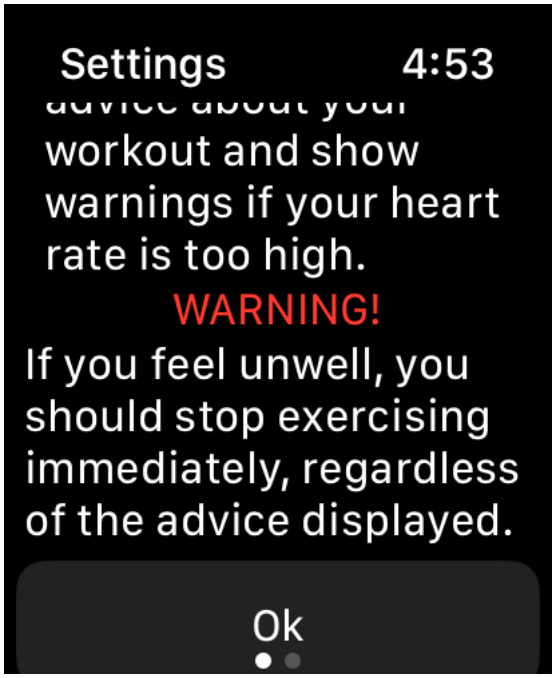


Рисунок 3.8 – Меню довідки з короткою інформацією про застосунок

В меню видалення даних тренування користувач отримує можливість видалити усі раніше зібрані дані тренувань. Для цього він повинен натиснути слайдер підтвердження вибору та натиснути кнопку «Delete» – рисунок 3.9.

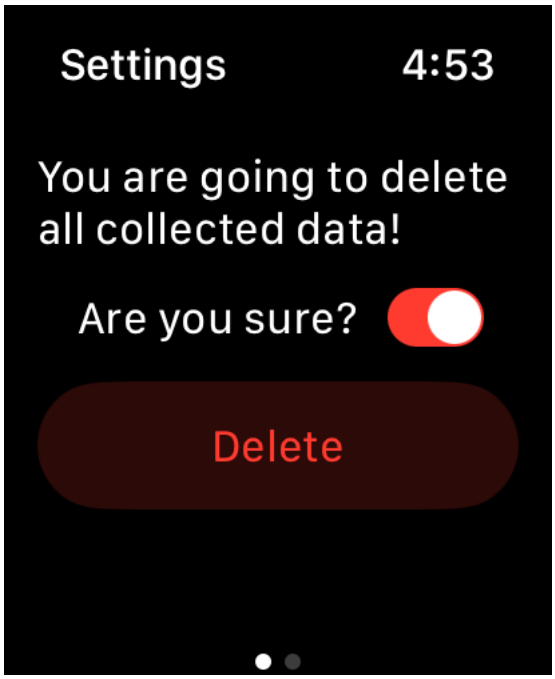


Рисунок 3.9 – Меню видалення даних тренувань з підтвердженням видалення

Якщо ж користувач відкрив меню, але не підтверджує видалення даних, буде активна лише кнопка «Cancel». Після натискання користувач потрапить в меню налаштувань – рисунок 3.10.

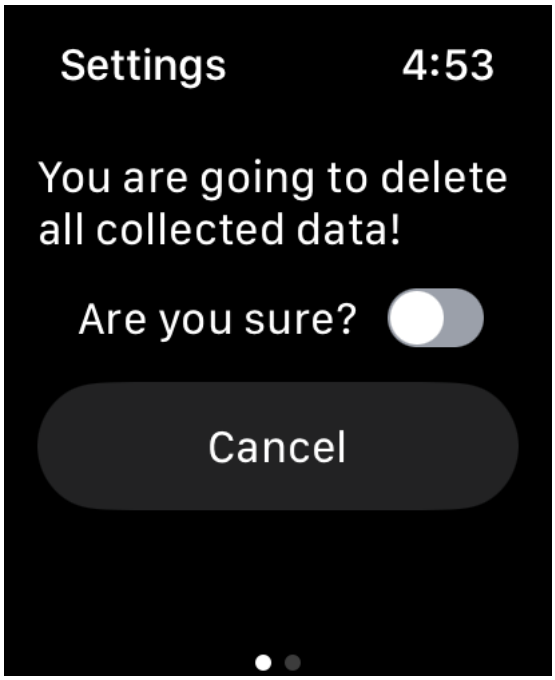


Рисунок 3.10 – Меню видалення даних тренувань без підтвердженням видалення

Після встановлення всіх налаштувань користувач переходить до меню вибору тренувань – рисунок 3.5.

Після вибору потрібного тренування користувач переходить в меню, де відображаються дані поточного тренування – рисунок 3.11.

Рисунок 3.11 – Меню відображення даних поточного тренування

Зі свайпом вниз користувач переходить до меню з відображенням порад та попереджень. Першу чверть встановленої дистанції користувач отримує пораду продовжити рухатись з таким самим темпом – рисунок 3.12.

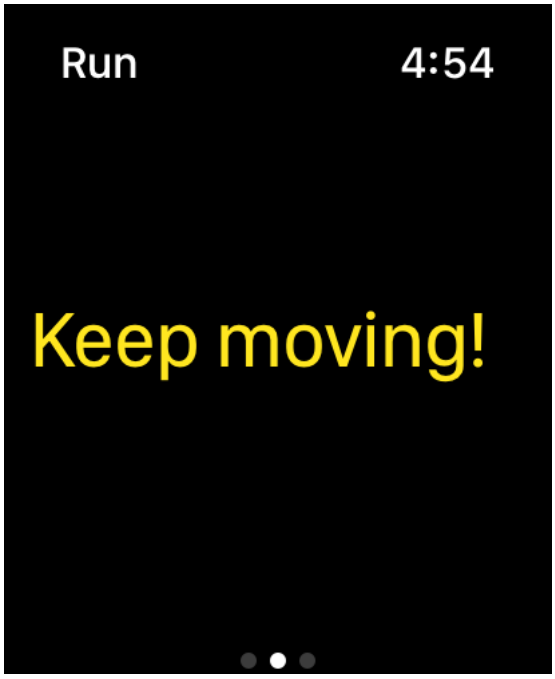


Рисунок 3.12 – Екран порад користувачу

Це необхідно для збору базової статистики для подальшого формування порад.

Також на цьому екрані користувач може бачити попередження про перевищення максимального рекомендованого пульсу.

Перед початком тренування було встановлено вік користувача як 99. Таким чином, програма буде показувати попередження при досягненні пульсу 121, розрахованого за формулою 220 мінус вік (99) – рисунок 3.13.

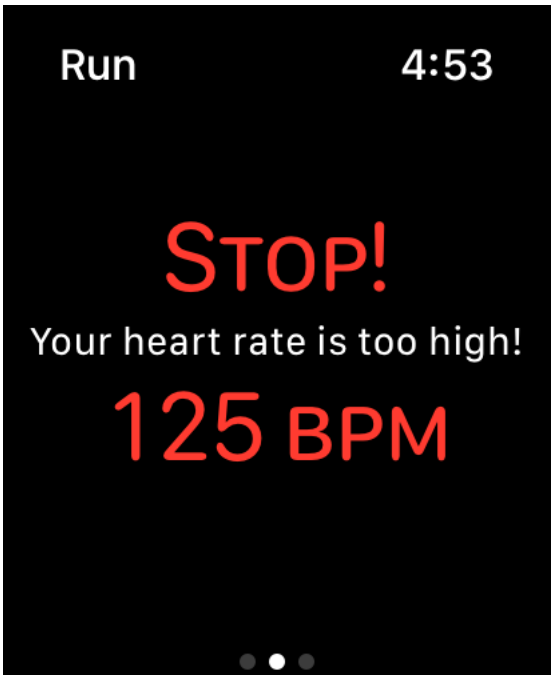


Рисунок 3.13 – Попередження користувача про досягнення максимального рекомендованого пульсу

Також користувач може свайпом вправо перейти до меню керування музикою – рисунок 3.14.

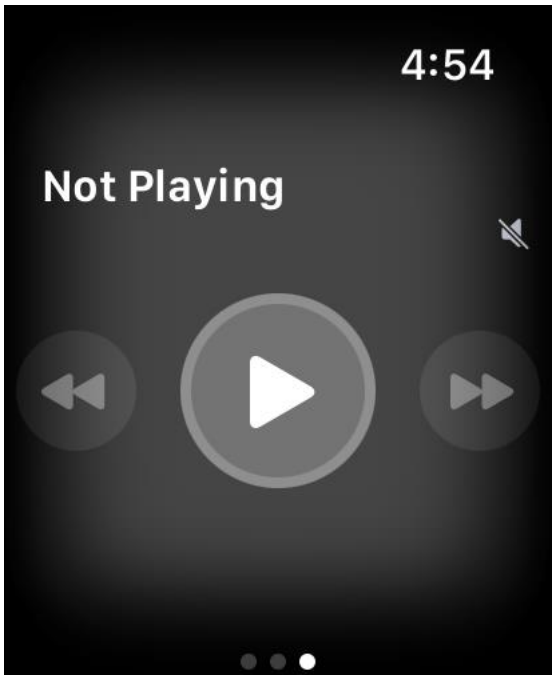


Рисунок 3.14 – Меню керування музикою

Слід зазначити, що музика має відігратися в сторонньому сервісі.

В застосунку наявна можливість встановити паузу поточного тренування. Для цього потрібно перейти в меню керування тренуванням і натиснути на кнопку «Pause» – рисунок 3.15.

Рисунок 3.15 – Встановлення паузи поточного тренування

Для зняття паузи та продовження тренування необхідно натиснути на цю кнопку ще раз, яка тепер вже має назву «Resume» – рисунок 3.16.

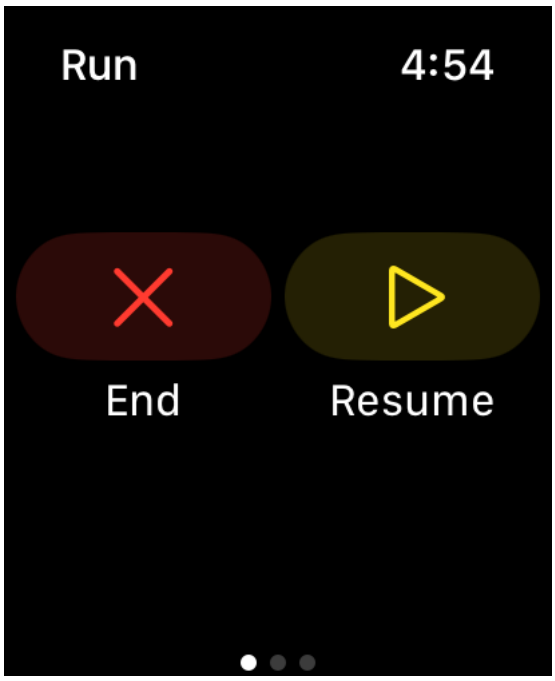


Рисунок 3.16 – Зняття паузи та продовження тренування

При закінченні тренування користувач має завершити його в застосунку, натиснувши кнопку «End».

Після цього користувач побачить зведену інформацію стосовно проведеного тренування – рисунок 3.17.

Рисунок 3.17 – Зведена інформація стосовно проведеного тренування

Також після завершення тренування буде додано в список усіх проведених тренувань на смартфоні. Користувач зможе його переглянути в застосунку Fitness, яке є стандартним для смартфонів від компанії Apple – рисунок 3.18.

Рисунок 3.18 – Відображення проведених тренувань в застосунку Fitness на смартфоні

Висновки до розділу

В цьому розділі дипломного проєкту було проаналізовано якість написаного коду, проведено мануальне тестування застосунку та описано увесь наявний функціонал застосунку у контрольному прикладі.

Для аналізу коду використано метод статичного аналізу. Він дозволяє проаналізувати якість написаного коду, без необхідності для аналізатора збирати та запускати проєкт. Такий аналіз дозволяє виявити потенційні проблеми та вразливості застосунку, а також відхилення від

загальноприйнятих практик написання коду. Останнє може не стосуватися працездатності і швидкодії застосунку, але дуже негативно впливає на розуміння написаного коду, а як наслідок – можливість підтримки та покращення застосунку. Для аналізу коду цього дипломного проєкту було обрано сервіс Embold. Він має інтеграцію з сервісами зберігання та контролю версій коду, такими як Github. Завдяки цьому є можливість проаналізувати проєкт онлайн, надавши доступ до потрібного репозиторія без необхідності розгортати сервіс локально на комп'ютері.

Для тестування проєкту було обрано мануальне тестування. Такий метод є доречним та оптимальним для цього проєкту, бо на момент тестування розробку усіх запланованих функцій було завершено і кожна з них тестувалася в процесі розробки. Таким чином, під час проведення мануального тестування потрібно було ще раз перевірити коректність роботи всіх функцій та алгоритмів і їх взаємодію.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для забезпечення користувачам можливості доступу до завантаження застосунку, існує лише один легальний, тобто дозволений компанією Apple спосіб. Для цього потрібно завантажити застосунок у магазин застосунків App Strore.

Першим кроком є створення профілю розробки та сертифікату розробки. Найпростішим способом зробити це є IDE XCode, в якому вже інтегрована функція автоматичного менеджменту профілів та сертифікатів.

Другим кроком є створення власного облікового запису App Store Connect – рисунок 4.1 [8].

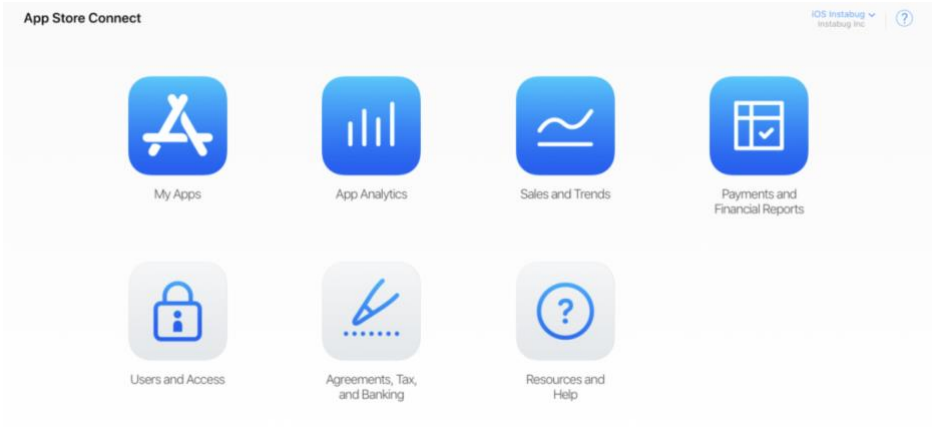


Рисунок 4.1 – Інформаційна панель App Store Connect

У ньому потрібно створити власну організацію і мати роль адміністратора, розробника або менеджера застосунків.

Якщо планується продавати застосунок, потрібно підписати договір, який охоплює умови оплати – рисунок 4.2 [8].

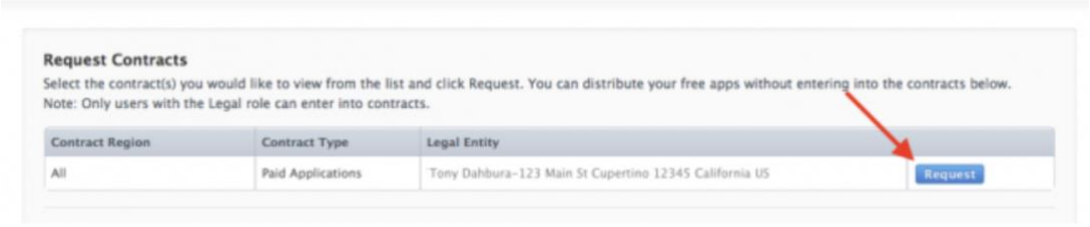


Рисунок 4.2 – Запит на договір про умови оплати

Після запиту на договір відобразиться інформація про умови оплати, з якою потрібно погодитись, поставивши відповідний прапорець.

Також потрібно додати контактну інформацію – рисунок 4.3 [8].

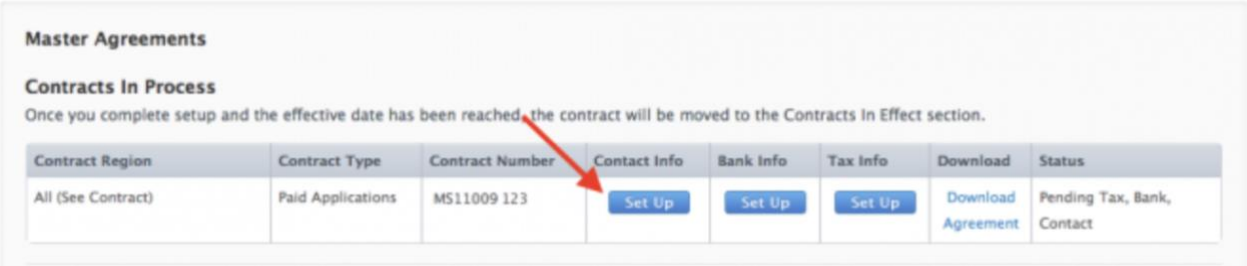


Рисунок 4.3 – Встановлення контактної інформації

У віконці, що з’явиться, потрібно натиснути кнопку «Додати новий контакт» та ввести необхідну інформацію. Також потрібно заповнити інформацію про банківський акаунт, податкову форму США, яка є обов’язковою та форму іншої країни, за необхідності. Після заповнення усієї необхідної інформації, статус контакту буде «Обробляється». Після обробки контакт з’явиться у розділі «Чинні контакти».

Наступним кроком є додавання застосунку. Для цього у App Store Connect потрібно обрати «Мої програми», натиснути на «+» у верхньому лівому куті, а потім на «Новий застосунок». Для створення нового запису потрібно буде заповнити назву застосунку, мову застосунку, платформу та унікальний ідентифікатор застосунку. Ці дані неможливо змінити пізніше, тому заповнювати слід уважно. Ідентифікатор застосунку має точно збігатися з ідентифікатором пакета у файлі Info.plist проєкту XCode.

Також потрібно зробити архів з програмою в застосунку XCode. У XCode необхідно обрати «Generic iOS Device» як ціль розгортання. Після обрання продукту у верхньому меню та натискання кнопки «Архів» запуститься XCode Organizer, у якому відобразяться будь-які архіви, які були створені раніше. Після обрання потрібного архіву потрібно натиснути кнопку «Завантажити в App Store», ввести свої облікові дані та натиснути кнопку «Обрати». У наступному вікні з’явиться кнопка завантаження і після завершення завантаження з’явиться повідомлення про успішне завершення.

Останнім кроком є заповнення решти необхідної інформації в App Store Connect. До неї відносяться додаткові мови, категорії, ціна завантаження, якщо застосунок не безкоштовний. Також потрібно додати інформацію, яка буде відображатися на головній сторінці застосунку в App Store. Це скріншоти з демонстрацією застосунку, короткий опис застосунку та вікове обмеження. Після завершення заповнення усієї інформації потрібно відправити застосунок на перевірку.

Перевірка застосунку може займати від одного до трьох днів і після успішної перевірки застосунок з'явиться в магазині застосунків [8].

4.2 Підтримка програмного забезпечення

Користувачі повинні мати можливість завантажувати нову та актуальну версію застосунку, або оновлювати свою поточну до актуальної.

Процес оновлення застосунку майже ідентичний до процесу першого завантаження застосунку в магазин App Store. У IDE XCode потрібно зробити новий архів з актуальною версією застосунку. У XCode Organizer потрібно обрати цей останній архів та натиснути «Завантажити в App Store». Після цього в App Store Connect потрібно оновити інформацію про нову версію, додати нові скріншоти застосунку за потреби і відправити на перевірку. Після завершення перевірки в магазині застосунків буде відображатися вже нова версія.

Висновки до розділу

У розділі було описано процес завантаження та оновлення застосунку в магазин застосунків App Store, бо це є єдиним легальним способом надати користувачам можливість завантажувати застосунок на свої девайси.

Для поширення застосунку через магазин застосунків App Store необхідно мати акаунт розробника, завдяки якому можливо використовувати App Store Connect. Перший раз потрібно заповнити профіль особистою контактною інформацією, даними про банківський акаунт та податковими

даними користувача. Після заповнення ці дані верифікуються і через деякий час з’являється можливість завантажити архів з застосунком.

Архівування застосунку виконується через IDE XCode, у якому і відбувалася розробка. Після цього окрім потрібного архіву слід заповнити інформацію про застосунок. Ці дані потім будуть відображатися на головному екрані застосунку в магазині застосунків. Після повної перевірки, яка займає від одного до 3х днів, застосунок можна буде завантажити безпосередньо в магазині застосунків.

Для оновлення версії застосунку в магазині застосунків App Store слід створити актуальний архів і аналогічно до першого завантаження надіслати його на перевірку, разом з коротким описом змін, присутніх в новій версії.

ВИСНОВКИ

В результаті виконання дипломного проєкту створено зручний додаток для розумних годинників, який буде аналізувати зібрану статистику тренувань та на її основі давати рекомендації по проведенню тренувань.

В якості середовища розробки обрано XCode, адже він є єдиним можливим варіантом розробки та тестування мобільних застосунків для девайсів від компанії Apple.

Для зберігання даних користувача використано комбінований підхід зі сховищем UserDefaults та БД SQLite. Сховище даних користувача добре підходить для зберігання малих обсягів інформації та містить лише два числа – вік та заплановану дистанцію тренування. БД містить багато даних про відрізки тренувань і надає можливість отримати певну послідовність даних використовуючи стандартні запити мовою SQL.

Після реалізації застосунку було перевірено коректне відображення інтерфейсу застосунку на усіх можливих за розмірами екрану пристроях. Завдяки використанню фреймворка SwiftUI, інтерфейс на девайсах з різним розміром екрану відображається однаково.

Новизна роботи полягає в реалізації алгоритму формування порад та застережень для користувача. Як було визначено при аналізі інших програмних продуктів, жоден застосунок не надає рекомендацій стосовно проведення поточного тренування і не застерігає користувача при досягненні високого пульсу, що може негативно вплинути на здоров’я користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Gregersen E. Smartwatch [Електронний ресурс] / Erik Gregersen – Режим доступу до ресурсу: https://www.britannica.com/technology/smartwatch.

2) SwiftUI [Електронний ресурс] – Режим доступу до ресурсу: https://developer.apple.com/XCode/swiftui/.

3) Robergs R. THE SURPRISING HISTORY OF THE “HRmax=220-age” / R. Robergs, R. Landwehr. – 2002. – С. 2–4.

4) HealthKit [Електронний ресурс] – Режим доступу до ресурсу: https://developer.apple.com/documentation/healthkit.

5) T. Dang A. MVC vs MVP vs MVVM [Електронний ресурс] / Anh T. Dang – Режим доступу до ресурсу: https://levelup.gitconnected.com/mvc-vs-mvp-vs-mvvm-35e0d4b933b4

6) Bellairs R. What Is Code Quality? [Електронний ресурс] / Richard Bellairs – Режим доступу до ресурсу: https://www.perforce.com/blog/sca/what-code-quality-overview-how-improve-code-quality.

7) Hamilton T. Manual Testing [Електронний ресурс] / Thomas Hamilton – Режим доступу до ресурсу: https://www.guru99.com/manual-testing.html.

8) Muscara A. How to Submit Your App to the App Store in 2023 [Електронний ресурс] / Aprille Muscara – Режим доступу до ресурсу: https://www.instabug.com/blog/how-to-submit-app-to-app-store.

ДОДАТКИ

ДОДАТОК А СЕРТИФІКАТ ПРО УЧАСТЬ У РОБОТІ МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

ЦФЕНД ЦЕНТР ФІНАНСОВО-ЕКОНОМІЧНИХ НАУКОВИХ ДОСЛІДЖЕНЬ

1865

CENTER FOR FINANCIAL-ECONOMIC RESEARCH
ЦЕНТР ФІНАНСОВО-ЕКОНОМІЧНИХ НАУКОВИХ ДОСЛІДЖЕНЬ

CERTIFICATE OF PARTICIPATION
СЕРТИФІКАТ УЧАСНИКА

№ 19-05-23-52

підтверджує, що

Щербаков Антон Олександрович

взяв участь у роботі Міжнародної науково-практичної конференції

«Наука, освіта та технології:
актуальні проблеми теорії та практики»

International scientific-practical conference

«Science, education and technology:
current problems of theory and practice»

Загальна кількість академічних годин: 6 год
(0,2 кредита ECTS)

Директор Центру фінансово-економічних наукових досліджень Щербак В. Д.

19 травня 2023 р.
May 19, 2023

м. Дрогобич, Україна
Drohobych, Ukraine

ДОДАТОК Б ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015345363

Дата перевірки:
31.05.2023 14:02:58 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
31.05.2023 14:05:00 EEST

ID користувача:
76913

Назва документа: ІТ-92 Щербаков ПЗ

Кількість сторінок: 74 **Кількість слів:** 10952 **Кількість символів:** 86711 **Розмір файлу:** 3.40 MB **ID файлу:** 1015013680

4.83% Схожість

Найбільша схожість: 1.88% з джерелом з Бібліотеки (ID файлу: 1014981446)

2.17% Джерела з Інтернету	131	Сторінка 76
4.6% Джерела з Бібліотеки	173	Сторінка 78

1.96% Цитат

Цитати	2	Сторінка 79

Не знайдено жодних посилань

0% Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 8 слів та 0%)

0% Вилучення з Інтернету	7	Сторінка 80
0% Вилученого тексту з Бібліотеки	6	Сторінка 80

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	78

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ЗАСТОСУНОК ДЛЯ РОЗУМНОГО ГОДИННИКА ДЛЯ МОНІТОРИНГУ АКТИВНОСТІ

Текст програми

КПІ.ІТ-9228.045430.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Антон ЩЕРБАКОВ

Київ – 2023

Файл AgeRequestView.swift

```
struct AgeRequestView: View {

    @State private var selectedAge: Int = UserDefaults.standard.integer(forKey:
ConstantsEnum.settingsNames.userAge)
    @State private var showAgeView1 = true
    @Environment(\.presentationMode) var presentationMode

    var body: some View {
        VStack(alignment: .center) {
            Text("Select your age")
                .font(.system(size: 20))
                .bold()
            Picker("", selection: $selectedAge, content: {
                        ForEach(0..<100) { age in
                            HStack {
                                Spacer()
                                Text("\(age)")
                                    .font(.system(size: 40))
                                    .tag(age)
                                Spacer()
                            }.padding()
                        }
                    })
                    .pickerStyle(WheelPickerStyle())
                    .frame(width: 120)
//              NavigationLink(destination: SettingsView(), isActive: $showAgeView1) {
                        Button(action: {
                            // Handle selected age here

                                UserDefaults.standard.setValue(selectedAge, forKey:
ConstantsEnum.settingsNames.userAge)
                                print(UserDefaults.standard.integer(forKey:
ConstantsEnum.settingsNames.userAge))

                            showAgeView1 = true
                            self.presentationMode.wrappedValue.dismiss()
```

```
                }) {
                    Text("Submit")
                }
            }

        .frame(minWidth: 0, maxWidth: .infinity, minHeight: 0, maxHeight: .infinity, alignment: .center)
    }
}

struct AgeRequestView_Previews: PreviewProvider {
    static var previews: some View {
        AgeRequestView()
    }
}
```

Файл DeleteDataView.swift

```
struct DeleteDataView: View {

    @State private var confirmedDeletion = false

    @State private var sliderValue: Double = 0
    @Environment(\.presentationMode) var presentationMode

    var body: some View {

        VStack {
            Text("You are going to delete all collected data!").padding()

            Toggle("Are you sure?",isOn: $confirmedDeletion )
                .toggleStyle(SwitchToggleStyle(tint: .red))
            .accentColor(confirmedDeletion ? .green : .red)
            .frame(width: 150)

            if (confirmedDeletion) {
                Button(action: {
                    if confirmedDeletion {
                        let db = DBManager()
```

```
                db.deleteData()
                print("deleteData")

                self.presentationMode.wrappedValue.dismiss()
            }
        }) {
            Text("Delete")
                .cornerRadius(10)
        }
        .disabled(!confirmedDeletion)
        .tint(.red)
        .padding()
    }
    else
    {
        Button(action: {
            if !confirmedDeletion {
                print("cancel")
                self.presentationMode.wrappedValue.dismiss()
            }
        }) {
            Text("Cancel")
                .cornerRadius(10)
        }
        .padding()
    }

  }
 }
}

struct DeleteDataView_Previews: PreviewProvider {
    static var previews: some View {
        DeleteDataView()
    }
}
```

Файл DistanceView.swift

```
struct DistanceView: View {
    @State private var selectedDistance: Int = UserDefaults.standard.integer(forKey:
ConstantsEnum.settingsNames.userDistance)
    @Environment(\.presentationMode) var presentationMode

    var body: some View {
        VStack(alignment: .center) {
                Text("Select distance (m)")
                    .font(.system(size: 20))
                    .bold()
                Picker("", selection: $selectedDistance, content: {
                    ForEach(1...1000, id: \.self) { distance in
                        let value = distance * 100
                        HStack {
                            Spacer()
                            Text("\(value)")
                                .font(.system(size: 25))
                                .tag(value)
                            Spacer()
                        }.padding()
                    }
                })
                    .pickerStyle(WheelPickerStyle())
                    .frame(width: 120)
            Button(action: {
                UserDefaults.standard.setValue(selectedDistance, forKey:
ConstantsEnum.settingsNames.userDistance)
                self.presentationMode.wrappedValue.dismiss()
            }) {
                Text("Submit")


            }
        }
        .frame(minWidth: 0, maxWidth: .infinity, minHeight: 0, maxHeight: .infinity, alignment: .center)
    }
}
```

```
struct DistanceView_Previews: PreviewProvider {
    static var previews: some View {
        DistanceView()
    }
}
```

Файл EnumConstants.swift

```
enum ConstantsEnum {

    enum settingsNames {
        static let userAge = "UserAge"
        static let userDistance = "UserDistance"
    }
    enum adviceNames {
        static let highHeartRate = "Your heart rate is too high!"
        static let keepMoving = "Keep moving!"
        static let reducePace = "To achieve the goal, it is better to reduce the pace!"
        static let increasePace = "You can pick up the pace a bit!"
    }
    enum info {
        static let info1 = "This app can display advice about your workout and show warnings if your heart rate is too high."
        static let infoWarning = "WARNING!"
        static let info2 = "If you feel unwell, you should stop exercising immediately, regardless of the advice displayed."
    }

}
```

Файл InfoView.swift

```
struct InfoView: View {
    @Environment(\.presentationMode) var presentationMode
    var body: some View {
        ScrollView {
            VStack(alignment: .center) {
```

```
            Text(ConstantsEnum.info.info1)
            Text(ConstantsEnum.info.infoWarning).foregroundColor(.red)
            Text(ConstantsEnum.info.info2)
            Button(action: {

                self.presentationMode.wrappedValue.dismiss()
            }) {
                Text("Ok")

            }
        }
    }
}

struct InfoView_Previews: PreviewProvider {
    static var previews: some View {
        InfoView()
    }
}
```

Файл LinearRegression.swift

```
class LinearRegression
{
    func linearRegression(x: [Double],  y: [Double], distance: Double) -> Double? {
        guard x.count == y.count else {
            return nil
        }

        let n = Double(x.count)
        let xMean = x.reduce(0, +) / n
        let yMean = y.reduce(0, +) / n

        var numerator = 0.0
        var denominator = 0.0

        for i in 0..<x.count {
```

```
            numerator += (x[i] - xMean) * (y[i] - yMean)
            denominator += (x[i] - xMean) * (x[i] - xMean)
        }

        guard denominator != 0 else {
            return nil // Return nil if denominator is zero
        }

        let slope = numerator / denominator
        let intercept = yMean - slope * xMean

        return slope * distance + intercept
    }

}
```

Файл MainView.swift

```
struct MainView: View {


    var body: some View {
        TimelineView(MetricsTimelineSchedule(from: trainingViewModel.dataB?.startDate ?? Date(),
isPaused: trainingViewModel.training?.state == .paused)) { context in

            TabView {
                        VStack(alignment: .leading) {
                TimeView(Time: trainingViewModel.dataB?.Time(at: context.date) ?? 0, showSubseconds:
context.cadence == .live)
                    .foregroundStyle(.yellow)

                // kilocalories.
                Text(Measurement(value: trainingViewModel.activeEnergy_training, unit:
UnitEnergy.kilocalories)
                        .formatted(.measurement(width: .abbreviated, usage: .workout, numberFormatStyle:
.number.precision(.fractionLength(0)))))

                // heart rate in beats per minute (bpm).
```

```
                Text(trainingViewModel.heartRate_training.formatted(.number.precision(.fractionLength(0)))
+ " bpm")
                    .foregroundStyle(.red)

                // distance in meters.
                Text(Measurement(value: trainingViewModel.distance_training, unit: UnitLength.meters)
                    .formatted(.measurement(width: .abbreviated, usage: .road)))
            }
            .font(.system(.title, design: .rounded).monospacedDigit().lowercaseSmallCaps())
            .frame(maxWidth: .infinity, alignment: .leading)
            .ignoresSafeArea(edges: .bottom)
            .scenePadding()
            .tabItem {
                Label("Metrics", systemImage: "")
            }
            VStack(alignment: .leading)  {
                if trainingViewModel.heartRate_training > (220 -
Double(UserDefaults.standard.integer(forKey: ConstantsEnum.settingsNames.userAge))) {
                    Text("Stop!")
                        .foregroundStyle(.red)
                        .frame(maxWidth: .infinity, alignment: .center)
                    Text(ConstantsEnum.adviceNames.highHeartRate)
                        .font(.system(size: 15))
                        .frame(maxWidth: .infinity, alignment: .center)

Text(trainingViewModel.heartRate_training.formatted(.number.precision(.fractionLength(0))) + " bpm")
                        .foregroundStyle(.red)
                        .frame(maxWidth: .infinity, alignment: .center)
                } else {
                    Text(trainingViewModel.currentAdvice)
                        .font(.system(size: 30))
                        .minimumScaleFactor(0.5)
                        .foregroundStyle(.yellow)
                }
            }
            .font(.system(.title, design: .rounded).monospacedDigit().lowercaseSmallCaps())
            .frame(maxWidth: .infinity, alignment: .leading)
```

```
                .ignoresSafeArea(edges: .bottom)
                .scenePadding()
                .tabItem {
                    Label("Text", systemImage: "")
                }
            }
            .tabViewStyle(.carousel)
        }
    }

    // Access the TrainingViewModel through the environment.
    @EnvironmentObject var trainingViewModel: TrainingViewModel

}

struct MetricsView_Previews: PreviewProvider {
    static var previews: some View {
        MainView().environmentObject(TrainingViewModel())
    }
}
```

Файл SettingsView.swift

```
struct SettingsView: View {
    @State private var showAgeView = false
    @State private var showDistanceView = false
    @State private var showInfoView = false
    @State private var showDeleteDataView = false

    var body: some View {
        NavigationView {
            VStack(spacing: 10) {
                //Text("Settings").font(.system(size: 20))
                HStack(spacing: 10) {

                    NavigationLink(destination: AgeRequestView(), isActive: $showAgeView) {
                        Button(action: {
                            showAgeView = true
```

```
        }) {
            Circle()
                .foregroundColor(.blue)
                .frame(width: 60, height: 60)
                .overlay(
                    Image(systemName: "person")
                        .foregroundColor(.white)
                        .font(.system(size: 30))
                )

        }.buttonStyle(PlainButtonStyle())
    }

    NavigationLink(destination: DistanceView(), isActive: $showDistanceView) {
        Button(action: {
            showDistanceView = true
        }) {
            Circle()
                .foregroundColor(.yellow)
                .frame(width: 60, height: 60)
                .overlay(
                    Image(systemName: "ruler")
                        .foregroundColor(.white)
                        .font(.system(size: 30))
                        .tint(.yellow)
                )
        }


    }.buttonStyle(PlainButtonStyle())
}.buttonStyle(PlainButtonStyle())

HStack(spacing: 10) {
    NavigationLink(destination: InfoView(), isActive: $showInfoView) {
        Button(action: {
            showInfoView = true
        }) {
```

```
                    Circle()
                        .foregroundColor(.green)
                        .frame(width: 60, height: 60)
                        .overlay(
                            Image(systemName: "questionmark")
                                .foregroundColor(.white)
                                .font(.system(size: 30))
                        )
                }
            }.buttonStyle(PlainButtonStyle())

            NavigationLink(destination: DeleteDataView(), isActive: $showDeleteDataView) {
                Button(action: {
                    showDeleteDataView = true
                }) {
                    Circle()
                        .foregroundColor(.red)
                        .frame(width: 60, height: 60)
                        .overlay(
                            Image(systemName: "xmark.bin")
                                .foregroundColor(.white)
                                .font(.system(size: 30))
                        )
                }
            }.buttonStyle(PlainButtonStyle())

        }.buttonStyle(PlainButtonStyle())
    }
    .padding()
    //.navigationBarTitle("Settings")

}.navigationBarTitle("Settings")

}
}
```

```
struct SettingsView_Previews: PreviewProvider {
    static var previews: some View {
        SettingsView()
    }
}
```

Файл SqLiteDB.swift

```
class DBManager
{
    init()
    {
        db = openDatabase()
        createTable()
        createTrainingSplitsTable()
    }


    let dbPath: String = "myDb.sqlite"
    var db:OpaquePointer?


    func openDatabase() -> OpaquePointer?
    {
        let filePath = try? FileManager.default.url(for: .documentDirectory, in: .userDomainMask,
appropriateFor: nil, create: false)
            .appendingPathComponent(dbPath)
        var db: OpaquePointer? = nil
        if sqlite3_open(filePath?.path, &db) != SQLITE_OK
        {
            debugPrint("can't open database")
            return nil
        }
        else
        {
            print("Successfully created connection to database at \(dbPath)")
            return db
        }
```

```
    }
    func deleteTable(name:String) {
        let createTableString = "Drop TABLE IF EXISTS \(name);"
        var createTableStatement: OpaquePointer? = nil
        if sqlite3_prepare_v2(db, createTableString, -1, &createTableStatement, nil) == SQLITE_OK
        {
            if sqlite3_step(createTableStatement) == SQLITE_DONE
            {
                print("\(name) table deleted.")
            } else {
                print("\(name) table could not be deleted.")
            }
        } else {
            print("DRop TABLE statement could not be prepared.")
        }
        sqlite3_finalize(createTableStatement)
    }
    func createTable() {
        let createTableString = "CREATE TABLE IF NOT EXISTS workout(Id INTEGER PRIMARY KEY,
date TEXT,type TEXT);"
        var createTableStatement: OpaquePointer? = nil
        if sqlite3_prepare_v2(db, createTableString, -1, &createTableStatement, nil) == SQLITE_OK
        {
            if sqlite3_step(createTableStatement) == SQLITE_DONE
            {
                print("workout table created.")
            } else {
                print("workout table could not be created.")
            }
        } else {
            print("CREATE TABLE statement could not be prepared.")
        }
        sqlite3_finalize(createTableStatement)
    }
    func createTrainingSplitsTable() {
```

```
        let createTableString = "CREATE TABLE IF NOT EXISTS trainingsplits(Id INTEGER PRIMARY
KEY,workoutId INTEGER, time TEXT,heartRate DOUBLE, activeEnergy DOUBLE, distance Double,
FOREIGN KEY(workoutId) REFERENCES workout(id));"
        var createTableStatement: OpaquePointer? = nil
        if sqlite3_prepare_v2(db, createTableString, -1, &createTableStatement, nil) == SQLITE_OK
        {
            if sqlite3_step(createTableStatement) == SQLITE_DONE
            {
                print("trainingsplits table created.")
            } else {
                print("trainingsplits table could not be created.")
            }
        } else {
            print("CREATE TABLE statement could not be prepared.")
        }
        sqlite3_finalize(createTableStatement)
    }


    func insert(id:Int, date:String, type:String)
    {
        let workouts = read()
        for w in workouts
        {
            if w.id == id
            {
                return
            }
        }
        let insertStatementString = "INSERT INTO workout (Id, date, type) VALUES (?, ?, ?);"
        var insertStatement: OpaquePointer? = nil
        if sqlite3_prepare_v2(db, insertStatementString, -1, &insertStatement, nil) == SQLITE_OK {
            sqlite3_bind_int(insertStatement, 1, Int32(id))

            sqlite3_bind_text(insertStatement, 2, (date as NSString).utf8String, -1, nil)
            sqlite3_bind_text(insertStatement, 3, (type as NSString).utf8String, -1, nil)
```

```
        if sqlite3_step(insertStatement) == SQLITE_DONE {
            print("Successfully inserted row.")
        } else {
            print("Could not insert row.")
        }
    } else {
        print("INSERT statement could not be prepared.")
    }
    sqlite3_finalize(insertStatement)
}
func insertTrainingSplits(id:Int, workoutId:Int, time:String, heartRate: Double, activeEnergy: Double,
distance: Double)
{
    let trainingSplits = readTrainingSplits()
    for t in trainingSplits
    {
        if t.id == id
        {
            return
        }
    }
    let insertStatementString = "INSERT INTO trainingsplits (Id,workoutId, time,heartRate, activeEnergy,
distance) VALUES (?, ?, ?, ?, ?, ?);"
    var insertStatement: OpaquePointer? = nil
    if sqlite3_prepare_v2(db, insertStatementString, -1, &insertStatement, nil) == SQLITE_OK {
        sqlite3_bind_int(insertStatement, 1, Int32(id))
        sqlite3_bind_int(insertStatement, 2, Int32(workoutId))
        sqlite3_bind_text(insertStatement, 3, (time as NSString).utf8String, -1, nil)
        sqlite3_bind_double(insertStatement, 4, Double(heartRate))
        sqlite3_bind_double(insertStatement, 5, Double(activeEnergy))
        sqlite3_bind_double(insertStatement, 6, Double(distance))

        if sqlite3_step(insertStatement) == SQLITE_DONE {
            print("Successfully inserted row.")
        } else {
            print("Could not insert row.")
        }
```

```
    } else {
        print("INSERT statement could not be prepared.")
    }
    sqlite3_finalize(insertStatement)
}

func read() -> [Workout] {
    let queryStatementString = "SELECT * FROM workout;"
    let (workoutSplits) = readWorkoutBody(queryStatementString: queryStatementString)
    return (workoutSplits)
}
func read(type: String) -> [Workout] {
    let queryStatementString = "SELECT * FROM workout where type = '\(type)';"
    let (workoutSplits) = readWorkoutBody(queryStatementString: queryStatementString)
    return (workoutSplits)

}
func readWorkoutBody(queryStatementString: String) -> ([Workout]){
    var queryStatement: OpaquePointer? = nil
    var trainingSplits : [Workout] = []
    if sqlite3_prepare_v2(db, queryStatementString, -1, &queryStatement, nil) == SQLITE_OK {
        //print("Query Result:")
        while sqlite3_step(queryStatement) == SQLITE_ROW {
            let id = sqlite3_column_int(queryStatement, 0)
            let date = String(describing: String(cString: sqlite3_column_text(queryStatement, 1)))
            let type = String(describing: String(cString: sqlite3_column_text(queryStatement, 2)))
            trainingSplits.append(Workout(id: Int(id), date: String(date), type: String(type)))
            //print("\(id) | \(date) | \(type)")
        }
    } else {
        print("SELECT statement could not be prepared")
    }
    sqlite3_finalize(queryStatement)
    return trainingSplits
}
func readTrainingSplits() -> [TrainingSplits] {
    let queryStatementString = "SELECT * FROM trainingsplits;"
```

```
        var queryStatement: OpaquePointer? = nil
        var trainingSplits : [TrainingSplits] = []
        if sqlite3_prepare_v2(db, queryStatementString, -1, &queryStatement, nil) == SQLITE_OK {
            //print("Query Result:")
            while sqlite3_step(queryStatement) == SQLITE_ROW {
                let id = sqlite3_column_int(queryStatement, 0)
                let workoutId = sqlite3_column_int(queryStatement, 1)
                let time = String(describing: String(cString: sqlite3_column_text(queryStatement, 2)))
                let heartRate = sqlite3_column_double(queryStatement, 3)
                let activeEnergy = sqlite3_column_double(queryStatement, 4)
                let distance = sqlite3_column_double(queryStatement, 5)
                trainingSplits.append(TrainingSplits(id:Int(id), workoutId:Int(workoutId), time:String(time),
heartRate: Double(heartRate), activeEnergy: Double(activeEnergy), distance: Double(distance)))
                //print("\(id) | \(workoutId)| \(time)| \(heartRate)| \(activeEnergy)| \(distance)")
            }
        } else {
            //print("SELECT statement could not be prepared")
        }
        sqlite3_finalize(queryStatement)
        return trainingSplits
    }

    func readTrainingSplits(worloutId: Int) -> ([TrainingSplits], [Double], [Double], [Double])  {
        let queryStatementString = "SELECT * FROM trainingsplits where workoutId = \(worloutId);"
        let (trainingSplits,activeEnergy, distance, heartRate) =
readTrainingSplitsBody(queryStatementString: queryStatementString)
        return (trainingSplits, activeEnergy, distance, heartRate)
    }
    func readTrainingSplits(type: String, Distance: Double) -> ([TrainingSplits], [Double], [Double],
[Double]) {
        let queryStatementString = "SELECT t.Id, t.workoutId, t.time, t.heartRate, t.activeEnergy, t.distance
FROM trainingsplits t JOIN workout w ON t.workoutId = w.id WHERE w.type = '\(type)' AND t.distance
BETWEEN \(Distance * 0.9) AND \(Distance * 1.1) ORDER BY t.distance;"
        let (trainingSplits,activeEnergy, distance, heartRate) =
readTrainingSplitsBody(queryStatementString: queryStatementString)
        return (trainingSplits, activeEnergy, distance, heartRate)
    }
```

```
    func readTrainingSplits(type: String, worloutId: Int, distance: Double) -> ([TrainingSplits], [Double], [Double], [Double]) {
        let queryStatementString = "SELECT t.Id, t.workoutId, t.time, t.heartRate, t.activeEnergy, t.distance FROM trainingsplits t JOIN workout w ON t.workoutId = w.id WHERE w.type = '\(type)' AND w.id = \(worloutId) and t.distance <= \(distance);"
        let (trainingSplits,activeEnergy, distance, heartRate) = readTrainingSplitsBody(queryStatementString: queryStatementString)
        return (trainingSplits, activeEnergy, distance, heartRate)
    }
    func readTrainingSplits(type: String) -> ([TrainingSplits], [Double], [Double], [Double]) {
        let queryStatementString = "SELECT t.Id, t.workoutId, t.time, t.heartRate, t.activeEnergy, t.distance FROM trainingsplits t JOIN workout w ON t.workoutId = w.id WHERE w.type = '\(type)' ORDER BY t.distance;"
        let (trainingSplits,activeEnergy, distance, heartRate) = readTrainingSplitsBody(queryStatementString: queryStatementString)
        return (trainingSplits, activeEnergy, distance, heartRate)
    }
    func readTrainingSplitsBody(queryStatementString: String) -> ([TrainingSplits], [Double], [Double], [Double]){
        var queryStatement: OpaquePointer? = nil
        var trainingSplits : [TrainingSplits] = []
        var activeEnergy: [Double] = []
        var distance: [Double] = []
         var heartRate: [Double] = []
        if sqlite3_prepare_v2(db, queryStatementString, -1, &queryStatement, nil) == SQLITE_OK {
            while sqlite3_step(queryStatement) == SQLITE_ROW {
                let id = sqlite3_column_int(queryStatement, 0)
                let workoutId = sqlite3_column_int(queryStatement, 1)
                let time = String(describing: String(cString: sqlite3_column_text(queryStatement, 2)))
                let heartRateValue = sqlite3_column_double(queryStatement, 3)
                let activeEnergyValue = sqlite3_column_double(queryStatement, 4)
                let distanceValue = sqlite3_column_double(queryStatement, 5)
                let trainingSplit = TrainingSplits(id:Int(id), workoutId:Int(workoutId), time:String(time), heartRate: Double(heartRateValue), activeEnergy: Double(activeEnergyValue), distance: Double(distanceValue))
                trainingSplits.append(trainingSplit)
                activeEnergy.append(trainingSplit.activeEnergy)
                distance.append(trainingSplit.distance)
```

```
            heartRate.append(trainingSplit.heartRate)
        }
    } else {
        //print("SELECT statement could not be prepared")
    }
    sqlite3_finalize(queryStatement)
    return (trainingSplits, activeEnergy, distance, heartRate)
}

func deleteData(){
    print("deleteData2")
    self.deleteTable(name: "trainingsplits")
    self.deleteTable(name: "workout")
}
func deleteRawByID(id:Int, deleteStatementStirng:String) {
        var deleteStatement: OpaquePointer? = nil
        if sqlite3_prepare_v2(db, deleteStatementStirng, -1, &deleteStatement, nil) == SQLITE_OK {
            sqlite3_bind_int(deleteStatement, 1, Int32(id))
            if sqlite3_step(deleteStatement) == SQLITE_DONE {
                print("Successfully deleted row.")
            } else {
                print("Could not delete row.")
            }
        } else {
            print("DELETE statement could not be prepared")
        }
        sqlite3_finalize(deleteStatement)
    }
func deleteRowByID(id:Int) {

    let deleteTrainingsplitsRec = "DELETE FROM trainingsplits WHERE workoutId = \(id);"
    deleteRawByID(id: id, deleteStatementStirng: deleteTrainingsplitsRec)
    let deleteWorkoutRec = "DELETE FROM workout WHERE Id = \(id);"
    deleteRawByID(id: id, deleteStatementStirng: deleteWorkoutRec)


    }
```

```
}
```

Файл TrainingSplitsModel.swift

```
class TrainingSplits
{

    var id: Int = 0
    var workoutId: Int = 0
    var time: String = ""
    var heartRate: Double = 0
    var activeEnergy: Double = 0
    var distance: Double = 0



    init(id:Int, workoutId:Int, time:String, heartRate: Double, activeEnergy: Double, distance: Double)
    {
        self.id = id
        self.workoutId = workoutId
        self.time = time
        self.heartRate = heartRate
        self.activeEnergy = activeEnergy
        self.distance = distance
    }
}
```

Файл TrainingViewModel.swift

```
class TrainingViewModel: NSObject, ObservableObject {

    var timer: Timer?
    let hkHealthStore = HKHealthStore()
    var training: HKWorkoutSession?
    var dataB: HKLiveWorkoutBuilder?
    var count: Int = 1
    var trainingId: Int = 0
    var trainingType: String = ""
    var workoutsNumber: Int = 0
```

```
var currentAdvice: String = ConstantsEnum.adviceNames.keepMoving
@Published var averageHeartRate_training: Double = 0
@Published var heartRate_training: Double = 0
@Published var activeEnergy_training: Double = 0
@Published var distance_training: Double = 0
//@Published var speed: Double = 0
@Published var workout: HKWorkout?
@Published var location_training: HKWorkoutRoute?
@Published var power_training: Double = 0
@Published var isActiveSummaryView: Bool = false {
    didSet {
        if isActiveSummaryView == false {
            workoutSessionManager?.newTraining()
        }
    }
}
@Published var running = false {
    didSet {
        if running {
            startTimer()
        } else {
            stopTimer()
        }
    }
}
private var workoutSessionManager: TrainingOperationsManager?

func startTimer() {
    timer?.invalidate()
    timer = Timer.scheduledTimer(withTimeInterval: 15, repeats: true) { [weak self] _ in
        self?.addTrainingSplit()

        if (self!.workoutsNumber > 3) {
            self?.predictEnergy()
        }
    }
}
```

```
// Stops the timer
func stopTimer() {
    timer?.invalidate()
    timer = nil
}

var activeTraining: HKWorkoutActivityType? {
    didSet {
        guard let selectedTraining = activeTraining else { return }
        startTraining(workoutType: selectedTraining)
    }
}

func startTraining(workoutType: HKWorkoutActivityType) {
    startTimer()
    self.trainingType = String(workoutType.name)

    let configuration = HKWorkoutConfiguration()
    configuration.activityType = workoutType
    configuration.locationType = .outdoor

    do {
        training = try HKWorkoutSession(healthStore: hkHealthStore, configuration: configuration)
        dataB = training?.associatedWorkoutBuilder()
    } catch {
        return
    }

    training?.delegate = self
    dataB?.delegate = self

    dataB?.dataSource = HKLiveWorkoutDataSource(healthStore: hkHealthStore,
                                                workoutConfiguration: configuration)

    let startDate = Date()
    training?.startActivity(with: startDate)
```

```
        dataB?.beginCollection(withStart: startDate) { (success, error) in
            let db = DBManager()
            let date = Date() // the current date
            let dateFormatter = DateFormatter()
            dateFormatter.dateFormat = "yyyy-MM-dd HH:mm:ss"

            var workouts = db.read(type: workoutType.name)
            self.workoutsNumber = workouts.count


            if (self.workoutsNumber > 100) {
                db.deleteRowByID(id: workouts[0].id)
            }

            workouts = db.read()
            let id = workouts.last?.id ?? 0
            self.trainingId = id + 1
            db.insert(id: self.trainingId, date: dateFormatter.string(from: date), type:
String(workoutType.name))
            self.workoutSessionManager = TraininigOperationsManager(training: self.training!, viewModel:
self)
        }
    }


    func togglePause() {
        if running == true {
            self.pause()
        } else {
            resume()
        }
    }


    func statisticsUpdate(_ statistics: HKStatistics?) {
        workoutSessionManager?.statisticsUpdate(statistics)
    }
```

```
    func pause() {
        workoutSessionManager?.pause()
    }


    func resume() {
        workoutSessionManager?.resume()
    }

    func endTraining() {
        workoutSessionManager?.endTraining()
    }

    func addTrainingSplit() {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "yyyy-MM-dd HH:mm:ss"
        let db = DBManager()
        let trainingSplits = db.readTrainingSplits()
        let trainingSplitId = trainingSplits.last?.id ?? 0
        db.insertTrainingSplits(id: trainingSplitId + 1, workoutId: self.trainingId, time:
dateFormatter.string(from: Date()), heartRate: self.heartRate_training, activeEnergy:
self.activeEnergy_training, distance: self.distance_training)
    }


    func predictEnergy() {
        let db = DBManager()
        let (_, activeEnergy, distance, heartRate) = db.readTrainingSplits(worloutId: self.trainingId)
        let lg = LinearRegression()
        var energy: Double

        let userDistance = Double(UserDefaults.standard.integer(forKey:
ConstantsEnum.settingsNames.userDistance) * 100)

        print(distance.last!)
        if distance.last! > Double(userDistance / 4) {
```

```
            energy = lg.linearRegression(x: distance, y: activeEnergy, distance: userDistance) ?? 0
            let (trainingSplits2, activeEnergy2, _, _) = db.readTrainingSplits(type: self.trainingType, Distance:
userDistance)
            if (activeEnergy2.count != 0) {
                let activeEnergyMaxTrainingId = activeEnergy2.firstIndex(of: activeEnergy2.max()!)!
                let (_, _, _, heartRateAvg) = db.readTrainingSplits(type: self.trainingType, worloutId:
trainingSplits2[activeEnergyMaxTrainingId].workoutId, distance: distance.last!)
                let sumHCurrent  = heartRate.reduce(0, +)
                _ = Double(sumHCurrent) / Double(heartRate.count)
                let sumHRecorder  = heartRateAvg.reduce(0, +)
                let meanHRecorder = Double(sumHRecorder) / Double(heartRateAvg.count)
                if (energy > activeEnergy2.max()! && sumHCurrent > meanHRecorder) {
                    print(ConstantsEnum.adviceNames.reducePace)
                    self.currentAdvice = ConstantsEnum.adviceNames.reducePace
                }
                if (energy < activeEnergy2.min()! && sumHCurrent < meanHRecorder) {
                    print(ConstantsEnum.adviceNames.increasePace)
                    self.currentAdvice = ConstantsEnum.adviceNames.increasePace
                } else {
                    print(ConstantsEnum.adviceNames.keepMoving)
                    self.currentAdvice = ConstantsEnum.adviceNames.keepMoving
                }
            } else {
                let (_, _, distanceMax, _) = db.readTrainingSplits(type: self.trainingType)
                energy =  lg.linearRegression(x: distance, y: activeEnergy, distance: distanceMax.max()!)!
                let (_, activeEnergy3, _, _) = db.readTrainingSplits(type: self.trainingType, Distance:
distanceMax.max()!)
                if (energy > activeEnergy3.max()! * 0.9) {
                    print(ConstantsEnum.adviceNames.reducePace)
                    self.currentAdvice = ConstantsEnum.adviceNames.reducePace
                } else {
                    print(ConstantsEnum.adviceNames.keepMoving)
                    self.currentAdvice = ConstantsEnum.adviceNames.keepMoving
                }
            }
        }
    }
```

```
}
```

Файл WorkoutModel.swift

```
class Workout
{

    var date: String = ""
    var id: Int = 0
    var type: String = ""

    init(id:Int, date:String, type:String)
    {
        self.id = id
        self.date = date
        self.type = type
    }
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ЗАСТОСУНОК ДЛЯ РОЗУМНОГО ГОДИННИКА ДЛЯ МОНІТОРИНГУ АКТИВНОСТІ

Програма та методика тестування

КПІ.ІТ-9228.045430.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Антон ЩЕРБАКОВ

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є застосунок для розумного годинника для аналізу фізичної активності. Застосунок було розроблено для платформи WatchOS.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

– перевірка збереження даних;

– перевірка видалення даних

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- динамічне тестування на відповідність функціональним вимогам;
- тестування на розумних годинниках з різними розмірами екрану;
- тестування на роботу з різними версіями операційної системи;
- тестування інтерфейсу користувача;
- тестування зручності використання;
- тестування алгоритму надання порад користувачу;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ЗАСТОСУНОК ДЛЯ РОЗУМНОГО ГОДИННИКА ДЛЯ МОНІТОРИНГУ АКТИВНОСТІ

Керівництво користувача

КПІ.ІТ-9228.045430.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Антон ЩЕРБАКОВ

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Розроблена програма - це застосунок для розумного годинника для аналізу фізичної активності. Кінцева збірка представляє собою повністю готовий до роботи застосунок, який може надавати користувачу рекомендації стосовно проведення його тренування і виводити застереження у разі досягнення максимального рекомендованого пульсу.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність мобільного пристрою з операційною системою на базі WatchOS з версією 9.0 або вище;

– для встановлення додатку на мобільному пристрої повинно бути не менше 50 МБ вільної пам’яті.

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч виконавши збирання проєкту в середовищі Xcode, під’єднавши смартфон Iphone, до якого під’єднаний розумний годинник.

2.3 Перевірка коректної роботи

По завершенню встановлення додатка на робочому столі мобільного пристрою повинна відобразитись іконка даного застосунку. У разі, якщо дана іконка не з’явилась, то встановлення відбулось не успішно. Інакше користувач має змогу запустити додаток, клацнувши на його іконку. Після натискання повинна відобразитись сторінка запиту на надання доступу до даних.

3 ВИКОНАННЯ ПРОГРАМИ

Після першого запуску застосунку користувач побачить повідомлення про необхідність надати застосунку доступ до даних здоров'я Health – рисунок 3.2

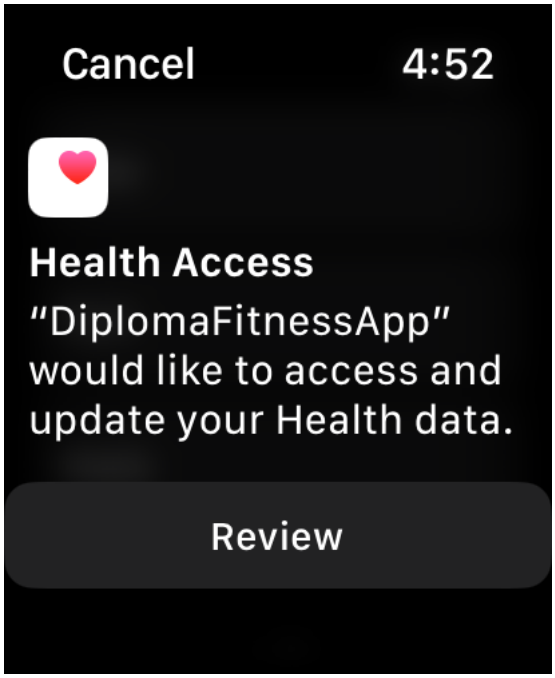


Рисунок 3.2 – Запит застосунку на доступ до даних

Після натискання кнопки "Review", у користувача з'явиться можливість надати окремо дозвіл на читання даних з застосунку Health та на запис даних – рисунки 3.3 - 3.4.

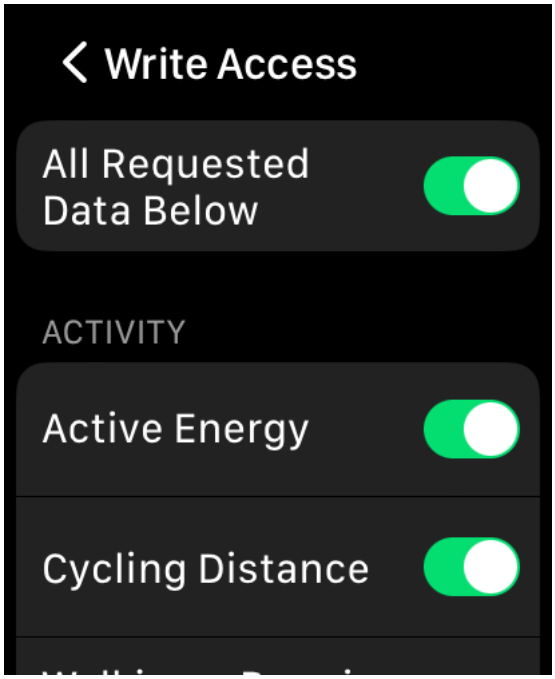


Рисунок 3.3 – Надання доступу на запис даних.

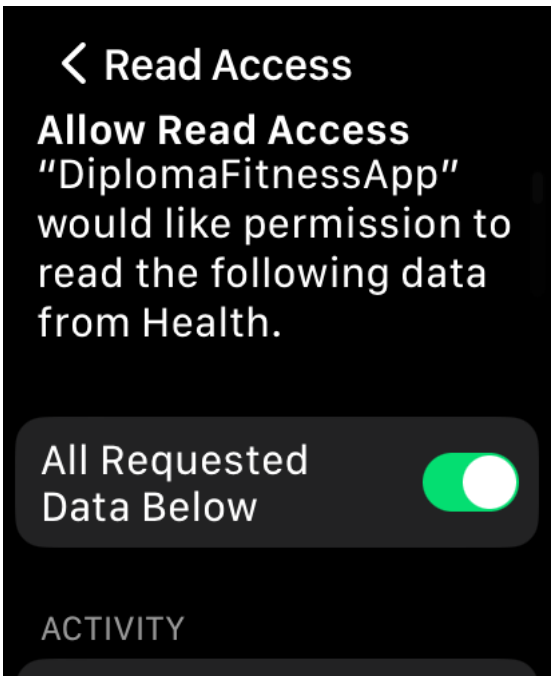


Рисунок 3.4 – Надання доступу на читання даних

Для коректної роботи застосунку необхідно надати доступ до всіх даних, про які запитує застосунок.

Після надання даних користувач потрапляє на головну сторінку вибору тренування – рисунок 3.5.

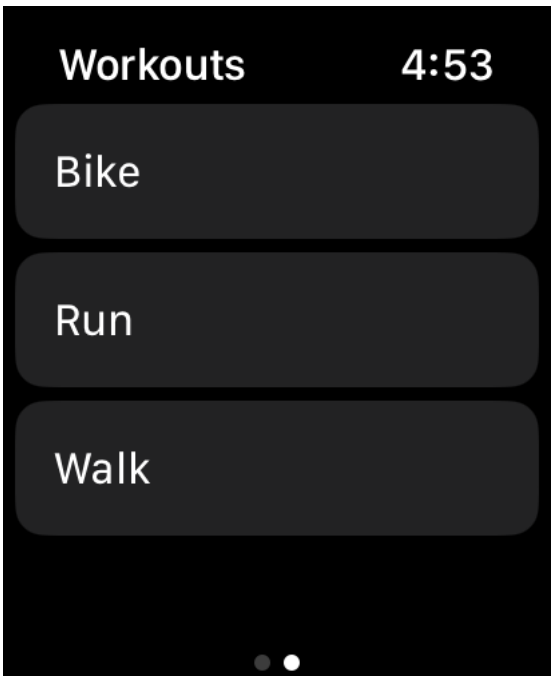


Рисунок 3.5 – Головне меню з наявними тренуваннями

Через те, що це перша взаємодія користувача з застосунком, перед початком тренування необхідно встановити налаштування користувача. Це

можна зробити, перейшовши в меню налаштувань лівим свайпом – рисунок 3.6.

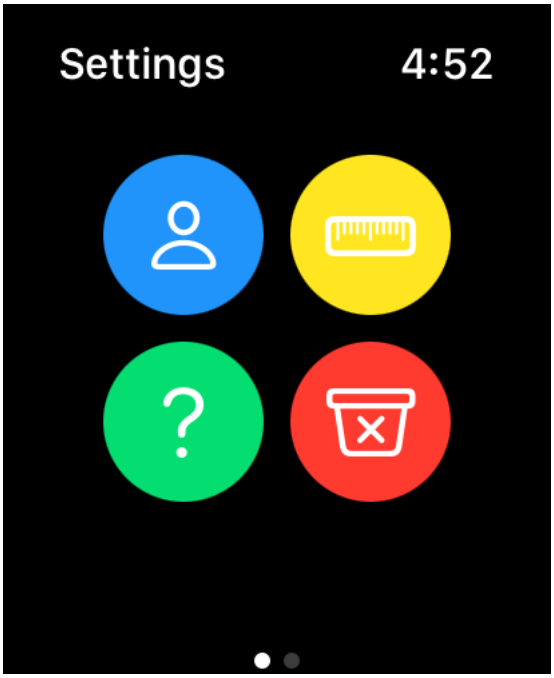


Рисунок 3.6 – Меню налаштувань застосунку

В меню доступні 4 кнопки:

– синя з іконкою людини – налаштування віку користувача;

– жовта з іконкою людини – налаштування віку користувача;

– зелена з іконкою питального знаки – отримання довідки про застосунок;

– червона з іконкою корзини – видалення всіх даних тренувань з бази даних.

Натиснувши синю кнопку відкриємо меню налаштування віку користувача – рисунок 3.7.

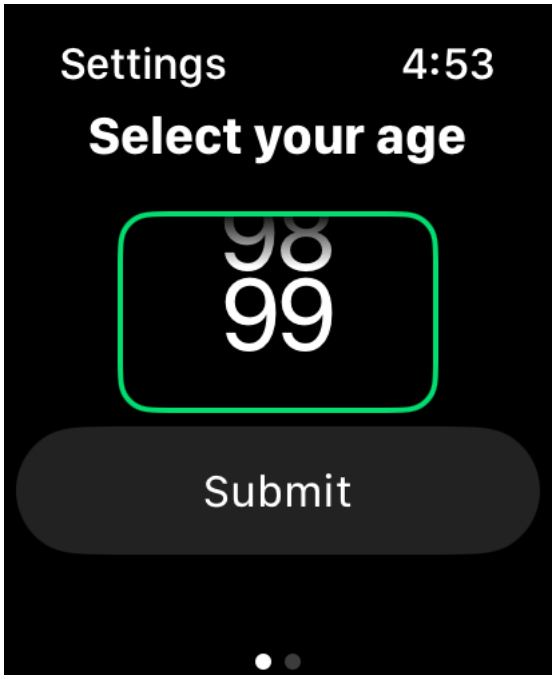


Рисунок 3.7 – Меню налаштування віку користувача

Після вибору потрібного параметру натискаємо кнопку «Submit», вік користувача зберігається в застосунку і користувач повертається до екрану налаштувань.

Аналогічно встановимо дистанцію тренування користувача – рисунок 3.8.

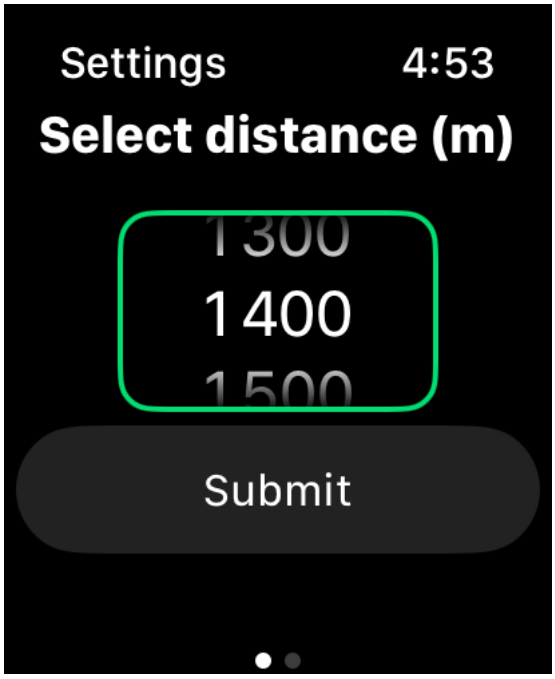


Рисунок 3.7 – Меню налаштування дистанції користувача

В меню отримання довідки користувач може прочитати коротку інформацію про застосунок – рисунок 3.8.

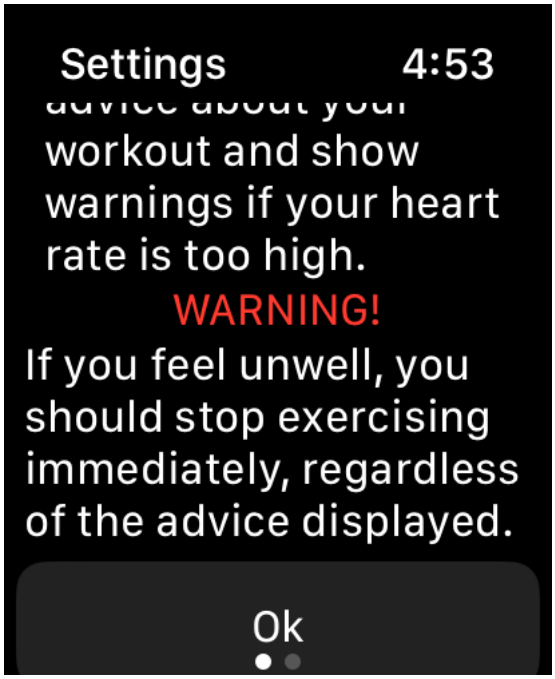


Рисунок 3.8 – Меню довідки з короткою інформацією про застосунок

В меню видалення даних тренування користувач отримує можливість видалити усі раніше зібрані дані тренувань. Для цього він повинен натиснути слайдер підтвердження вибору та натиснути кнопку «Delete» – рисунок 3.9.

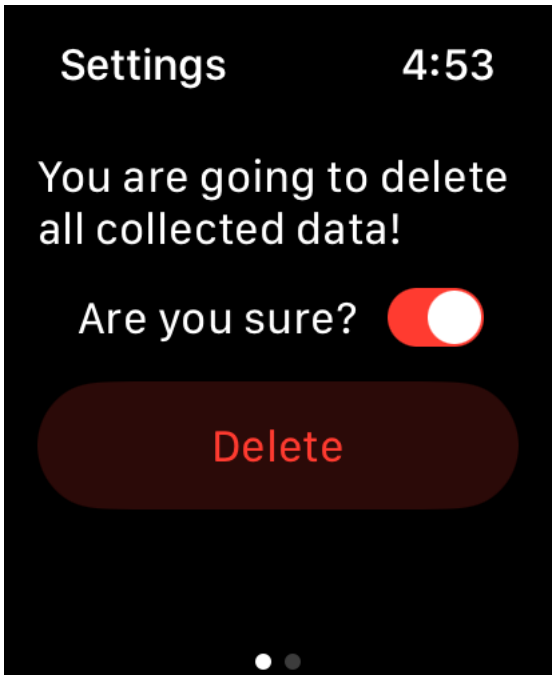


Рисунок 3.9 – Меню видалення даних тренувань з підтвердженням видалення

Якщо ж користувач відкрив меню, але не підтверджує видалення даних, буде активна лише кнопка «Cancel». Після натискання користувач потрапить в меню налаштувань – рисунок 3.10.

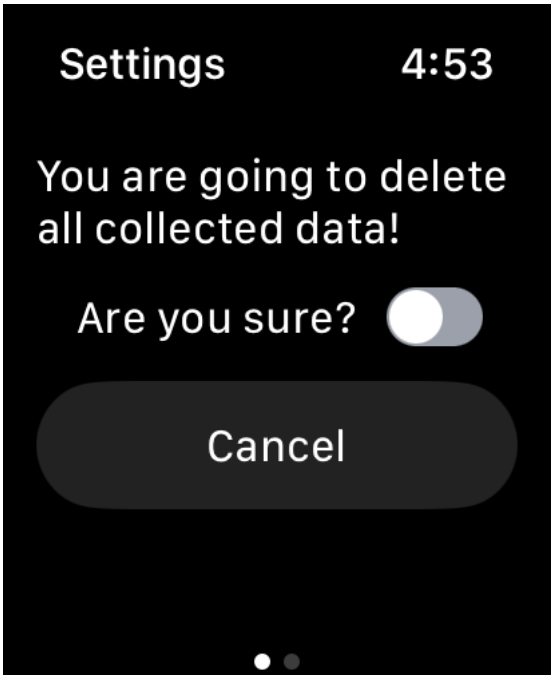


Рисунок 3.10 – Меню видалення даних тренувань без підтвердженням видалення

Після встановлення всіх налаштувань користувач переходить до меню вибору тренувань – рисунок 3.5.

Після вибору потрібного тренування користувач переходить в меню, де відображаються дані поточного тренування – рисунок 3.11.

Рисунок 3.11 – Меню відображення даних поточного тренування

Зі свайпом вниз користувач переходить до меню з відображенням порад та попереджень. Першу чверть встановленої дистанції користувач отримує пораду продовжити рухатись з таким самим темпом – рисунок 3.12.

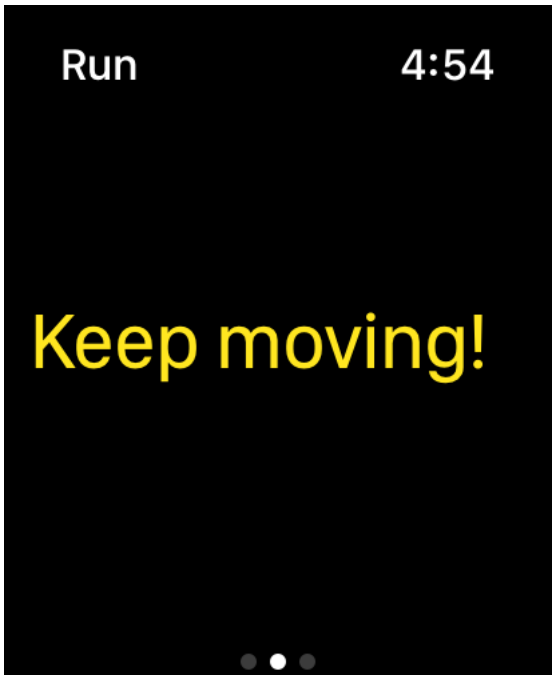


Рисунок 3.12 – Екран порад користувачу

Це необхідно для збору базової статистики для подальшого формування порад.

Також на цьому екрані користувач може бачити попередження про перевищення максимального рекомендованого пульсу.

Перед початком тренування було встановлено вік користувача як 99. Таким чином, програма буде показувати попередження при досягненні пульсу 121, розрахованого за формулою 220 мінус вік (99) – рисунок 3.13.

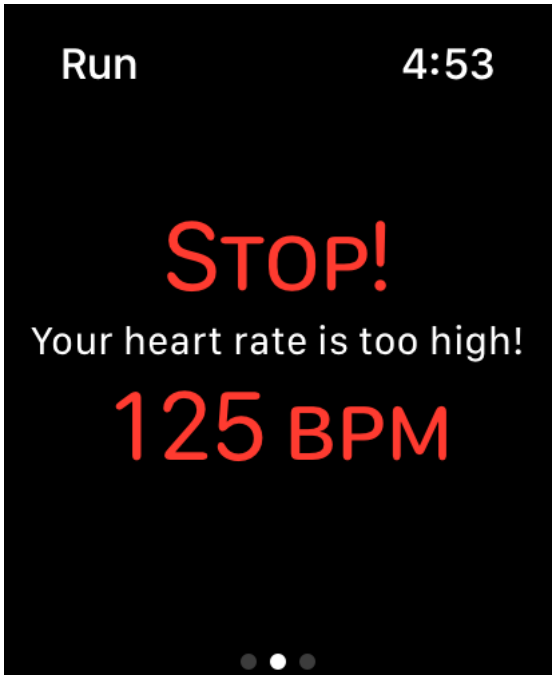


Рисунок 3.13 – Попередження користувача про досягнення максимального рекомендованого пульсу

Також користувач може свайпом вправо перейти до меню керування музикою – рисунок 3.14.

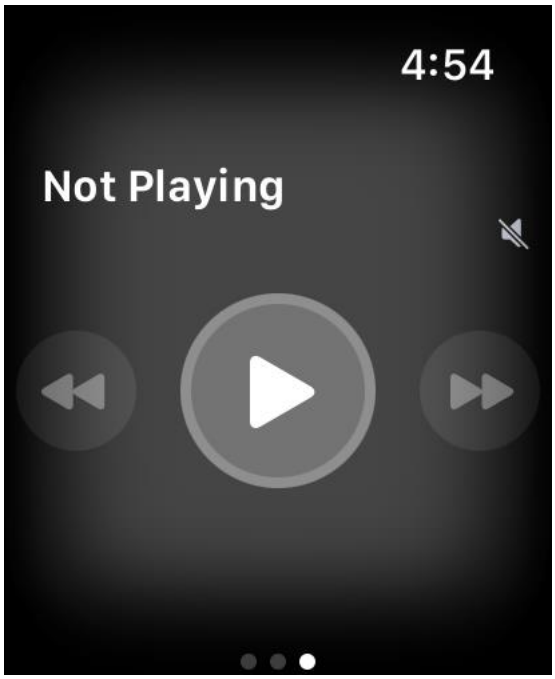


Рисунок 3.14 – Меню керування музикою

Слід зазначити, що музика має відіграватися в сторонньому сервісі.

В застосунку наявна можливість встановити паузу поточного тренування. Для цього потрібно перейти в меню керування тренуванням і натиснути на кнопку «Pause» – рисунок 3.15.

Рисунок 3.15 – Встановлення паузи поточного тренування

Для зняття паузи та продовження тренування необхідно натиснути на цю кнопку ще раз, яка тепер вже має назву «Resume» – рисунок 3.16.

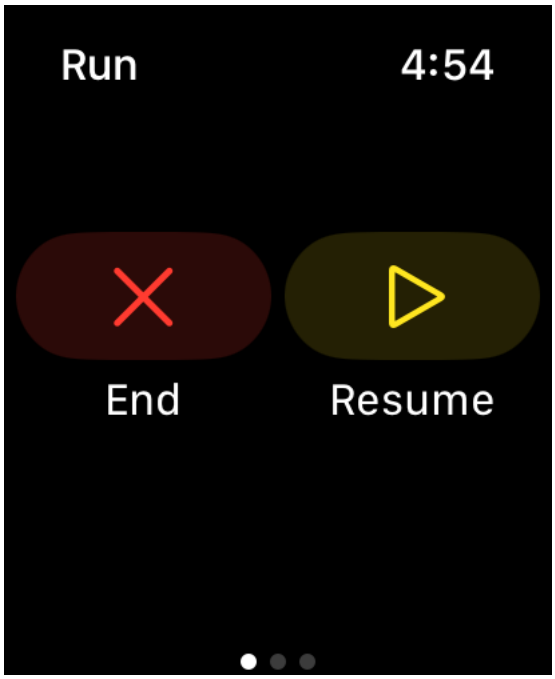


Рисунок 3.16 – Зняття паузи та продовження тренування

При закінченні тренування користувач має завершити його в застосунку, натиснувши кнопку «End».

Після цього користувач побачить зведену інформацію стосовно проведеного тренування – рисунок 3.17.

Рисунок 3.17 – Зведена інформація стосовно проведеного тренування

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ЗАСТОСУНОК ДЛЯ РОЗУМНОГО ГОДИННИКА ДЛЯ МОНІТОРИНГУ АКТИВНОСТІ

Графічний матеріал

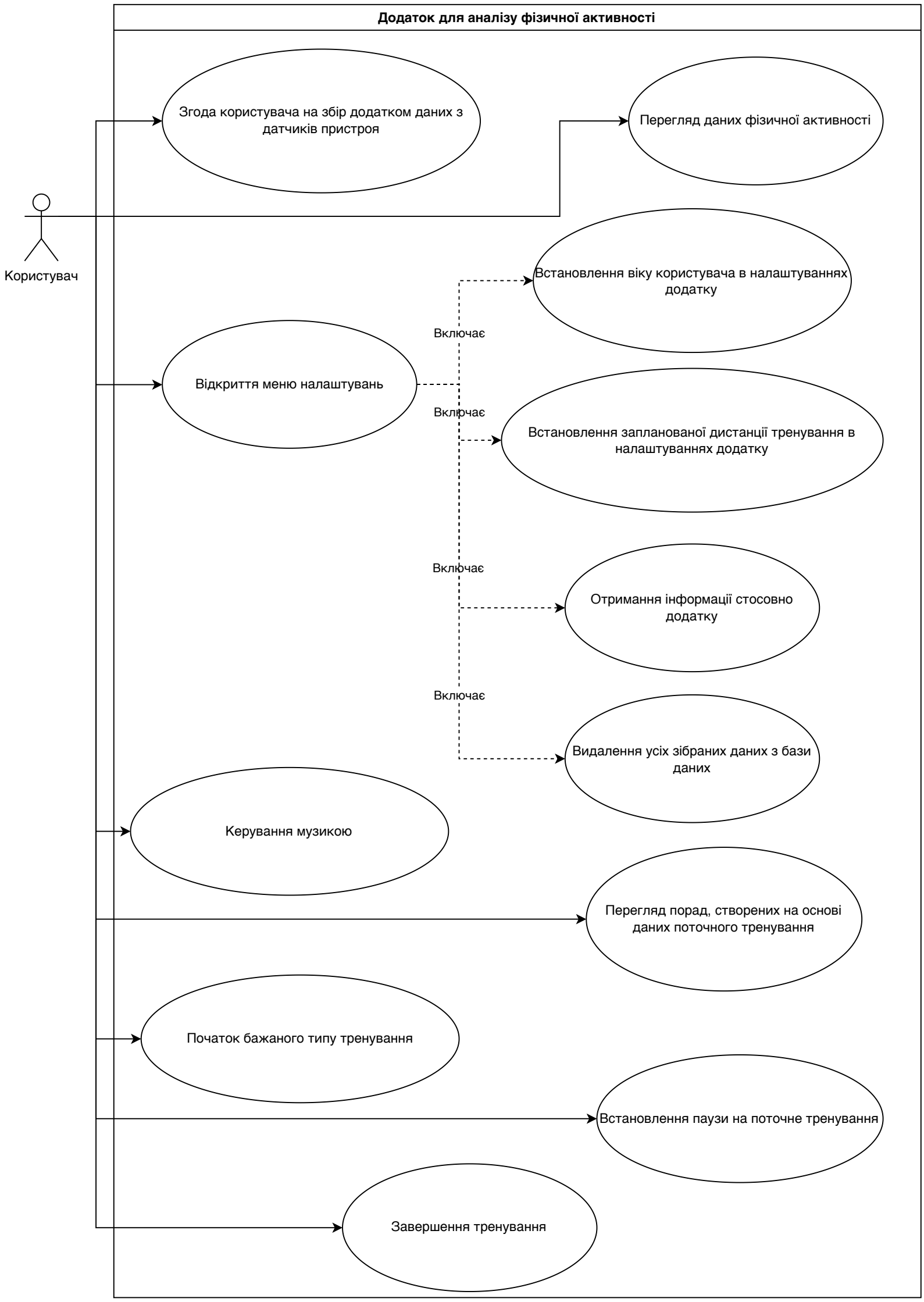
КПІ.ІТ-9228.045430.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

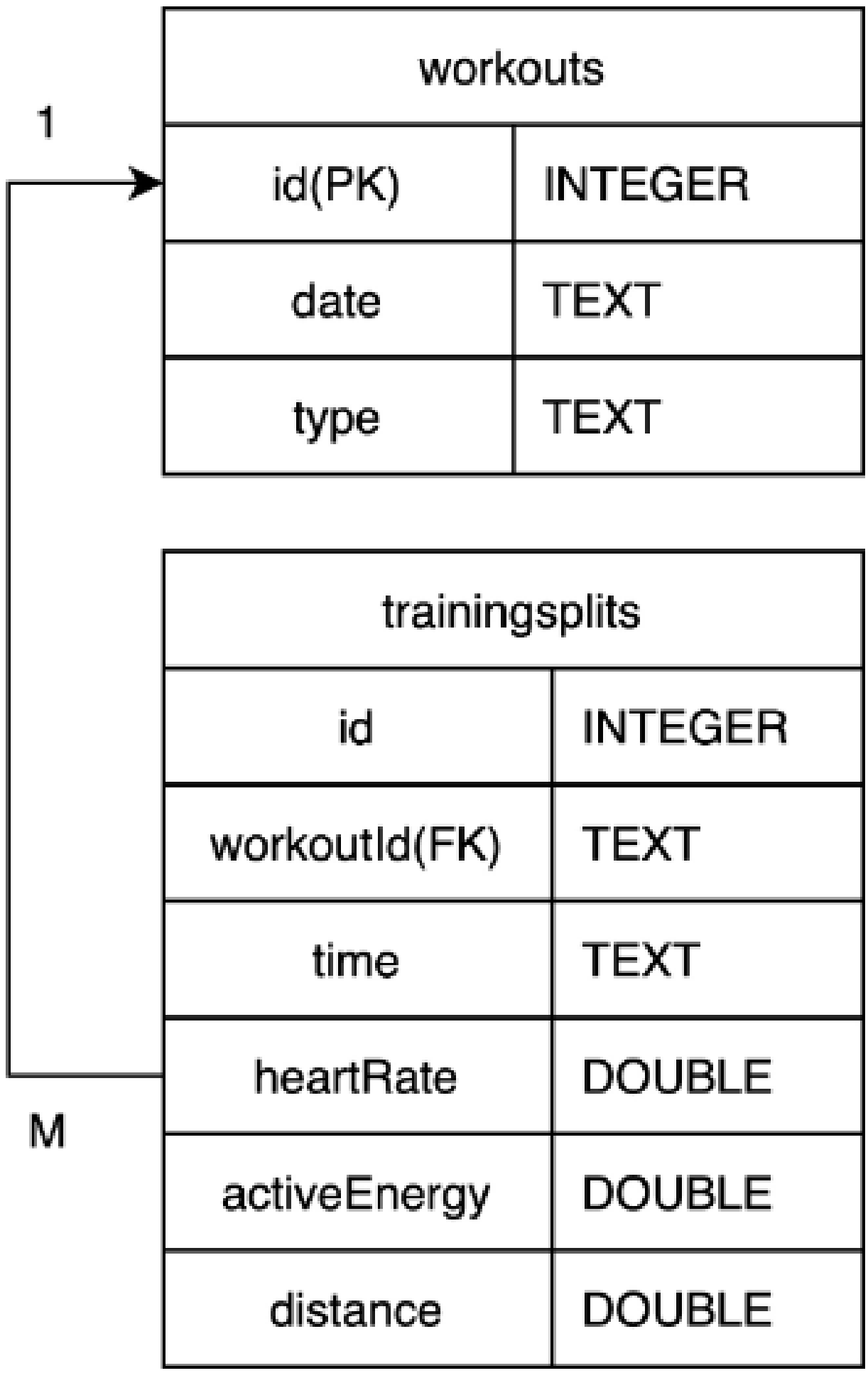
Виконавець:
_____ Антон ЩЕРБАКОВ

Київ – 2023

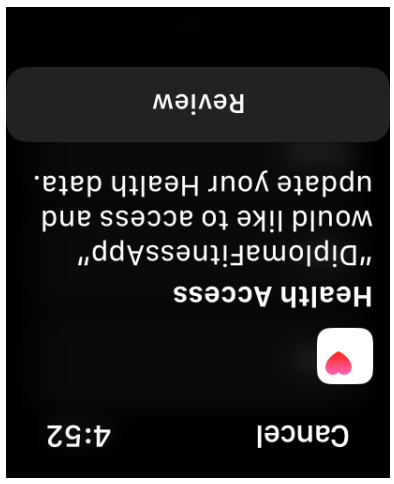


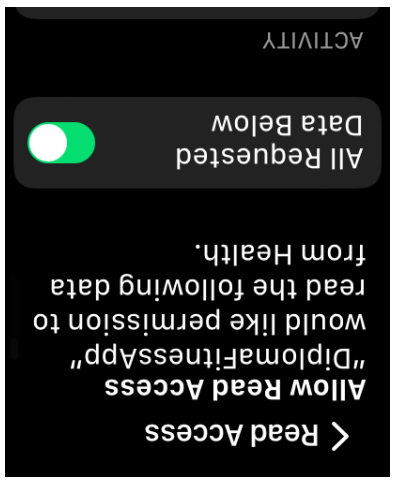
					КПІ.ІТ-9228.045430.06.99.ССВ	
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна варіантів використання	Лит. / Арк. / Аркушів: 1 / Аркуш 1 / Аркушів 4
Розроб.		Щербаков А.О.				
Перев.		Ліхоузова Т.А.				
Т. Кон.						
Н. Кон.		Вітковська І.І.			Застосунок для розумного годинника для моніторингу активності	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-92
Затв.		Жаріков Е.В.				

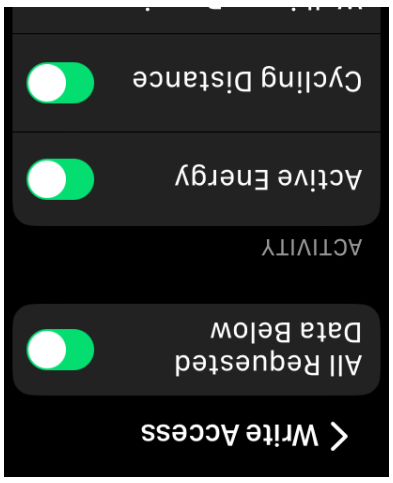
КПІ.ІТ-9228.045430.06.99.СБД

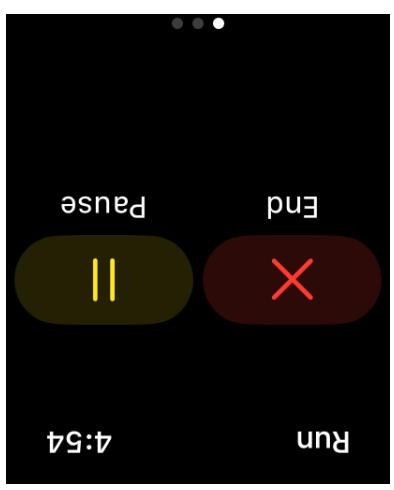


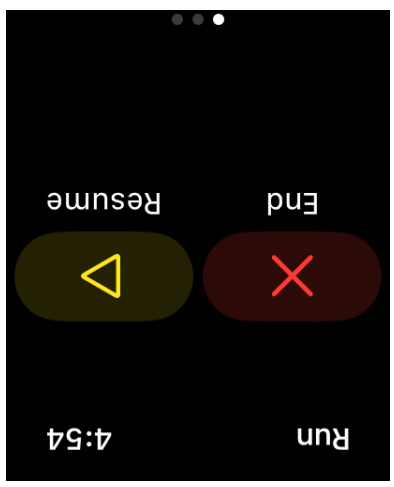
					КПІ.ІТ-9228.045430.06.99.СБД
Зм.	Арк.	№ докум.	Підп.	Дата	Схема бази даних
Розроб.		Щербаков А.О.			Лит. / Арк. / Аркушів: 1
Перев.		Ліхоузова Т.А.			Аркуш 2 / Аркушів 4
Т. Кон.					Застосунок для розумного годинника для моніторингу активності
Н. Кон.		Вітковська І.І.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-92
Затв.		Жаріков Е.В.			

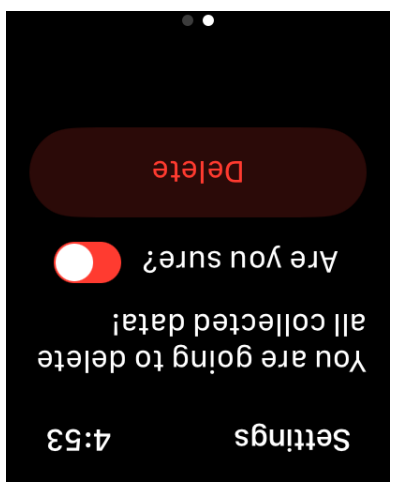


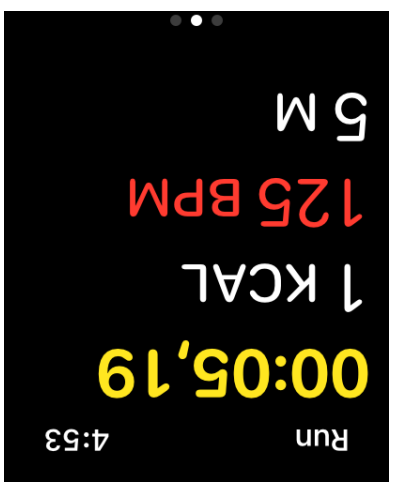












Зм.	Арк.	№ докум.	Підп.	Дата
Розроб.		Шевдаков А.О.		
Перев.		Лихоузова Т.А.		
Т. Кон.				
Н. Кон.		Бітковська І.І.		
Затв.		Жарікова Е.В.		

КПІ.ІТ-9228.045430.06.99.КЕ

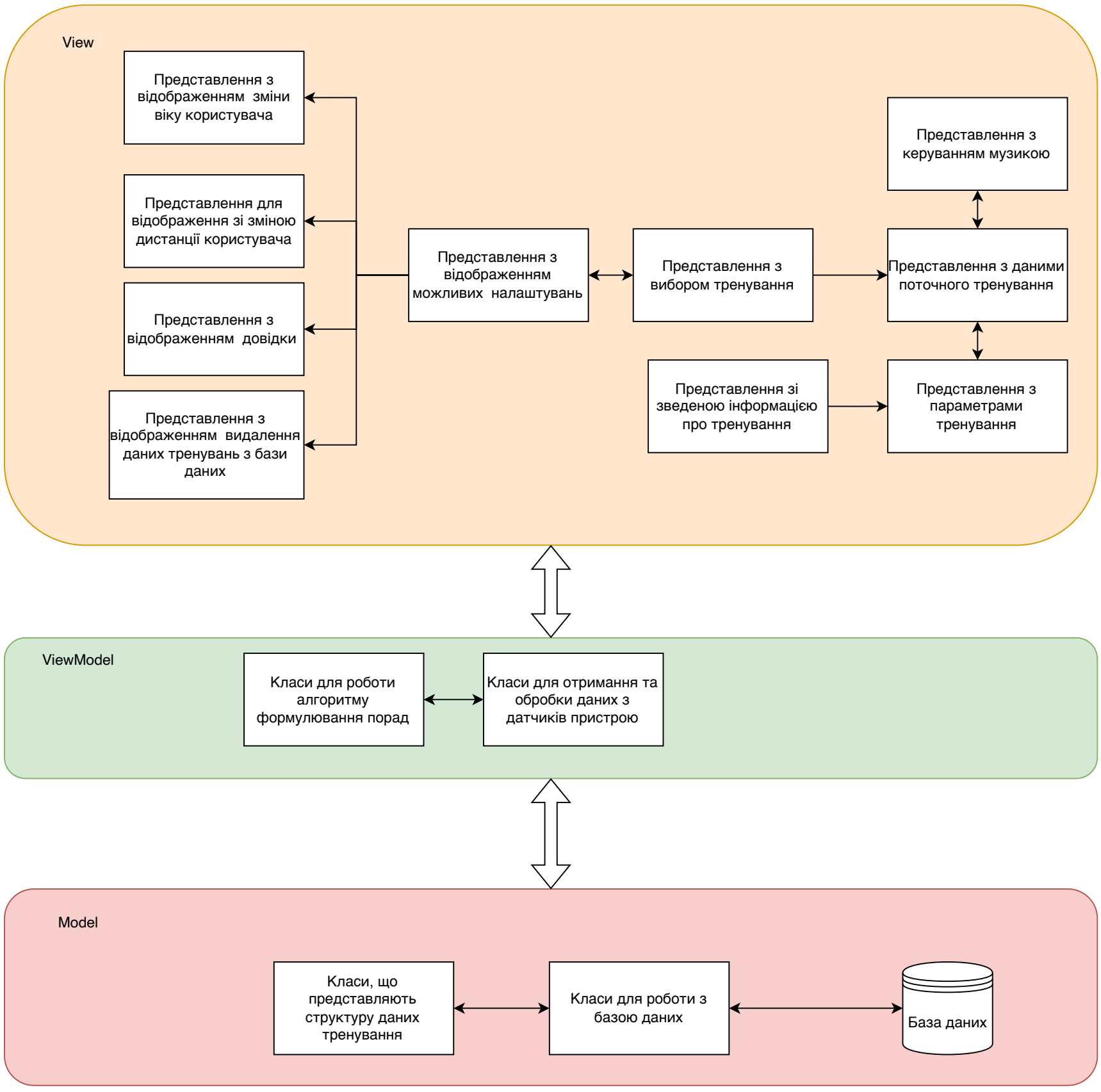
Літ.	Арк.	Аркушів
		1

Кресленння екранних форм

Аркуш 3 | Аркушів 4

Засоскунок для розумного годинника для моніторингу активності

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-92



					КПІ.ІТ-9228.045430.06.99.ССМ
Зм.	Арк.	№ докум.	Підп.	Дата	
Розроб.		Щербаков А.О.			Схема структурна компонентів програмного забезпечення
Перев.		Ліхоузова Т.А.			
Т. Кон.					
Н. Кон.		Вітковська І.І.			Застосунок для розумного годинника для моніторингу активності
Затв.		Жаріков Е.В.			

Лит.	Арк.	Аркушів
		1
Аркуш 4		Аркушів 4

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-92