Software Requirements Specification
for
MEDRhythms Mobile App
Version 1.0

Prepared by

Name	Date Signed
Ifeanyi Ineh	16/03/2025
Yiran Zhao	16/03/2025
Yogaa Srinivas Kassireddy	16/03/2025
Chaoyi Jiang	16/03/2025

Group Name:Fantastic Four

Instructor: Dr. Gary Cantrell

Course: Foundations of Software Engineering

Teaching Assistant: Sam Morris

March 16, 2025

Contents

1 Revision History

Version	Date	Description	Primary Author
1.0	March 16, 2025	Initial Draft	Ifeanyi Ineh

2 Introduction

2.1 Purpose

This document outlines the software requirements specification (SRS) for the MedRhythms App, which will allow users to monitor and check their step goals, predict fatigue levels for MVP[1], as well as integrate with music services like Spotify, Apple Music for V1 [2] and store this data securely using Firebase, This integration could continue towards WatchOS by using Flutter to build an App for SmartWatch to identify further features like fatigue etc. - V2 [3].

2.2 Document Conventions

This document follows IEEE formatting standards, including single-spacing, 1-inch margins, and use of Times New Roman font size 11 or 12.

2.3 Client and Stakeholders

2.3.1 Client

The client for this project is **MedRhythms**, a company specializing in developing therapeutic solutions for individuals with neurological conditions and stroke survivors. Their mission is to positively impact the lives of those living with neurologic injury and disease by building next-generation neurotherapeutics that leverage the power of music and technology to redefine what's possible for patients around the world.

2.3.2 Stakeholders

The key stakeholders for the MedRhythms App project include:

2.3.3 Development Team

- **Role**: Responsible for designing, developing, and maintaining the app.
- **Interests**:
  - Making sure the app is functional.
  - Delivering the project on time.
  - Writing clean, maintainable, and well-documented code.
  - Maintain the application database.

2.3.4 Project Manager

- **Role**: Oversees the app planning, execution, and delivery.
- **Interests**:
  - Ensuring the project stays on schedule and within the current scope.
  - Communicating progress to all stakeholders.

2.3.5 Testers (Quality Assurance Team)

- **Role**: Ensures the app is free of bugs and meets quality standards.
- **Interests**:
  - Identifying and reporting issues in the code and app.
  - Making sure that the app meets functional and non-functional requirements.
  - Deliver a product with a minimum number of bugs

[1] Minimum Viable Product
[2] V1 - Version Release 1.
[3] V2 - Version Release 2.

2.3.6 End-Users

- **Role**: The primary users of the app include individuals with neurological conditions, Parkinson's disease, stroke survivors, and their caregivers.
- **Interests**:
  - A user-friendly interface for ease of access.
  - Accurate tracking of physical activity (steps taken, distance covered, speed, heart rate).
  - Easy integration with music services such as Spotify.
  - Data privacy and security.

2.4 Intended Audience

The intended audience includes software developers, project manager, Quality Assurance Engineer, client, and other end-users who are interested in understanding the functionalities, requirements, and behavior of our product.

2.5 Scope

The **MedRhythms Mobile App** is designed to assist patients with stroke, parkinson's disease, as well as other neurological problems in monitoring and managing their walking performance through step tracking and music playback that aligns with their walking rhythm.

2.5.1 In Scope

The following functionalities will be included in the system:

- **User Authentication:**
  - Log in using a unique **IMEI number**.
  - Secure storage of login data through **Firebase authentication**.
- **Health Tracking & Data Collection:**
  - Step counting, distance tracking, heart rate and speed monitoring.
  - Session-based data processing and real-time statistics generation.
  - Synchronization with **Apple Health Kit(iOS)** and **Google Fit (Android)**.
  - Integration with **Samsung Health** as a priority.
- **Music Integration:**
  - **Spotify integration** to enhance user workouts.
  - Music playback that aligns with the walking rhythm.
- **Data Storage & Security:**
  - Cloud-based data storage using **Firebase**.
  - Local data caching for offline functionality with autosync upon reconnection.
  - **End-to-end encryption** for sensitive data.

2.5.2 Out of Scope

The following functionalities are **not included** in the current version (V1) of the app:

- Direct medical diagnostics like predicting health conditions beyond fatigue estimation.
- Wearable device support beyond mobile-integrated sensors (no smartwatches in V1).
- Multiplatform desktop/web versions (only available for iOS and Android).
- Customizable user profiles beyond basic settings.
- Support for multiple user accounts on the same device.
- No background tracking without user consent.

2.5.3 System Constraints & Boundaries

- The app will be developed using the **Flutter framework** for **cross-platform** support.
- **Minimum OS requirements:**
  - Android **13.0+**
  - iOS **13.0+**
- The system **requires an active internet connection** for cloud data synchronization but will store local data for offline access.
- **Performance Target:** The app should launch within **2 seconds** on supported devices.
- The system will require permission tp read health data from the user.

2.6 Definitions, Acronyms, and Abbreviations

- SRS – Software Requirements Specification
- IMEI - International Mobile Equipment Identity
- Firebase - Cloud-based backend service for authentication, database, and storage
- HealthKit – Apple's health-tracking API
- Google Fit – Google's health-tracking API
- MEDRhythms – Client
- IEEE – Institute of Electrical and Electronics Engineers
- GPS - Global Positioning System
- V1 - Version 1
- V2 - Version 2
- MVP - Minimum Viable Product

2.7 References

- IEEE SRS Standard 830-1998
- Firebase Documentation: `https://firebase.google.com/docs`
- Flutter Documentation: `https://flutter.dev/docs`
- Apple HealthKit Documentation: `https://developer.apple.com/documentation/healthkit`
- Google Fit API Documentation: `https://developers.google.com/fit`

3 Overall Description

3.1 Product Perspective

The MEDRhythms App is a mobile application that integrates with mobile health kits as well as other third-party APIs. It will store user data in Firebase for real-time synchronization.

3.2 Product Functions

- IMEI Login
- Collects Users' walk data
- Collects users' speed in mph
- Stores users' data in cloud DB

3.3 Constraints and Operating Environment

- Platforms: Android and iOS
- Backend: Firebase Cloud Firestore
- Frontend: Flutter

3.4 User Documentation Requirements

User documentation will include a user manual, FAQs, and troubleshooting guides.

3.5 Assumptions and Dependencies

3.5.1 Assumptions

- The app will be used primarily by individuals tracking their health and are clients of MedRhythms.
- Users will have compatible smartphones (Samsung) with sensors capable of tracking steps and motion.
- Internet connection is required for real-time Firebase synchronization, but the app should work offline as well with the data syncing when reconnected.
- Users are aware of their proprietary MedRhythms sensor IMEI number to enable them to log into the application.
- The battery of their device should be sufficiently charged for long-term tracking.

3.5.2 Dependencies

The mobile app depends on the following external systems, frameworks, and APIs:

- Firebase:
  - Cloud Firestore: To store user health tracking data
- Google Fit API & Apple HealthKit:
  - This will be required for step tracking and other health metrics integration.
- Health Connect for Android:
  - Health Connect: Allows users to store and share data between health and fitness apps.
- Flutter SDK:
  - The app will be developed using the Flutter framework for cross-platform support.
- Device Sensors:
  - Accelerometer & Gyroscope: Used for step counting and speed.

    * GPS
- Operating System:
    * The app will support Android 13.0+ and iOS 13+ to ensure compatibility with Firebase as well as Google Fit and Apple HealthKit APIs.

4 Functional Requirements

4.1 User Authentication

- **Description**: The app authenticates users based on their IMEI number.
- **Priority**: High
- **Stimulus**: The user launches the app for the first time.
- **Response**: The app retrieves the IMEI and logs in the user.
- **Functional Requirements**:
  * The app shall provide an option for the user to input their IMEI number upon launch.
  * The system shall log in the user after they input their IMEI.
  * The app shall securely store IMEI records in Firebase Firestore.

4.2 Step Tracking

- **Description**: The app tracks the user's steps, distance, and speed.
- **Priority**: High
- **Stimulus**: The user starts a session and begins walking.
- **Response**: The app detects movement and updates the step counter.
- **Functional Requirements**:
  * The app shall use accelerometer and gyroscope sensors for step tracking.
  * The app shall sync step count data with Firebase.
  * The app shall notify users when they have walked for 30 minutes.

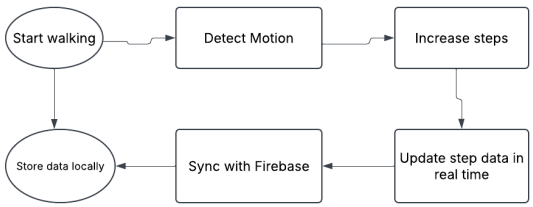

Figure 1: Activity Diagram for Step Tracking.

4.3 Music Integration

- **Description**: The app allows users to control music playback and play music.
- **Priority**: Medium
- **Stimulus**: The user starts a workout session.
- **Response**: The app starts playing music based on the user's step rhythm.
- **Functional Requirements**:
  * The app shall provide playback controls for music.
  * Users shall be able to connect their preferred music account (e.g., Spotify).

5 Use Cases

5.1 UC1 - Count Steps

- **Author**: Yoga Srinivas Reddy Kasireddy
- **Purpose**: To track and count the number of steps a user takes during a session.
- **Priority**: High
- **Stimulus**: The user starts walking or decides to record the session data.
- **Response**: The system detects motion and increments the step count.
- **Preconditions**:
  - The user has installed the app and granted necessary permissions.
- **Postconditions**:
  - Workout data is processed and stored.
  - If the device is online, data is uploaded to Firestore.
- **Actors**:
  - User – Walks and interacts with the app.
  - System – Detects steps and stores data.
- **Flow of Events**:
  - **Basic Flow**:
    1. User starts walking/decides to record the session data.
    2. The system detects motion sensor activity.
    3. The system increments the step count.
    4. Step count is stored locally and updated in real-time.
    5. If online, step data is uploaded to Firestore.
  - **Alternative Flow**:
    - If motion sensor data is delayed, steps are recorded in batches.
  - **Exceptions**:
    - If the motion sensor is disabled, the system prompts the user to enable it.
- **Notes/Issues**:
  - Needs calibration for different walking speeds and device types.

5.2 UC2 - Calculate Distance

- **Author**: Yoga Srinivas Reddy Kasireddy
- **Purpose**: To calculate the approximate distance traveled based on step count.
- **Priority**: Medium
- **Stimulus**: The user starts walking.
- **Response**: The system calculates and displays the distance walked.
- **Preconditions**:
  - Step tracking is active.
  - The user has granted necessary permissions.
- **Postconditions**:
  - Distance is calculated and stored for session tracking.
  - Data is uploaded to Firestore.
- **Actors**:
  - User – Walks.
  - System – Converts steps to distance and processes it.
- **Flow of Events**:

    - **Basic Flow**:
        1. User starts walking.
        2. Step count is tracked.
        3. System calculates the distance walked.
        4. Distance is displayed to the user and stored.
    - **Exceptions**:
        - If the sensor is unavailable, the system notifies the user to try again later.
- **Notes/Issues**:
    - Needs optimization for different walking patterns (e.g., running, jogging).

5.3 UC3 - Calculate Speed

- **Author**: Yoga Srinivas Reddy Kasireddy
- **Purpose**: To calculate the speed of movement based on step count and time.
- **Priority**: Medium
- **Stimulus**: The user starts moving at varying speeds.
- **Response**: The system calculates and displays the speed in real-time.
- **Preconditions**:
    - Step and distance tracking are active.
    - Time tracking is enabled.
- **Postconditions**:
    - Speed is calculated and displayed to the user.
- **Actors**:
    - User – Moves at varying speeds.
    - System – Computes speed from step count and time.
- **Flow of Events**:
    - **Basic Flow**:
        1. The system continuously records time and step count/distance.
        2. Speed is calculated as distance/time.
        3. Speed is displayed and stored.
    - **Alternative Flow**:
        - If the user stops, their speed becomes zero.
    - **Exceptions**:
        - If GPS is enabled, the system may use GPS speed instead.
- **Notes/Issues**:
    - May need testing to get accurate speed details.

5.4 UC4 - Music Integration (V1)

- **Author**: Yoga Srinivas Reddy Kasireddy
- **Purpose**: To integrate Spotify and play music based on user workouts or conditions.
- **Priority**: Medium
- **Stimulus**: The user starts a workout session and requests music playback.
- **Response**: The app connects to Spotify and starts playing music based on the user's step count or preferences.
- **Requirements Traceability**:
    - The user must have allowed Spotify integration.
    - The user must have configured their playlist settings.

    * The app must connect to Spotify and play music.
- **Preconditions**:
    * The user has a Spotify account and has granted access to the app.
    * The user has started a workout session.
- **Postconditions**:
    * Music is playing based on the user's step count or selected playlist.
    * The app is connected to Spotify.
- **Actors**:
    * **User** – Starts music and interacts with the app.
    * **System** – Connects to Spotify and controls music playback.
    * **Spotify API** – Provides music and playlist data.
- **Extends**: None
- **Flow of Events**:
    * **Basic Flow**:
        1. The user starts a workout session and requests music playback.
        2. The app connects to the Spotify API.
        3. The app retrieves the user's preferred playlist or selects a playlist based on step count.
        4. The app starts playing music from the selected playlist.
        5. The user listens to music while working out.
    * **Alternative Flow**:
        · The user selects a custom playlist instead of the default playlist.
    * **Exceptions**:
        · If the Spotify API is down, the app notifies the user and does not play music.
        · If the user revokes access to Spotify, the app prompts the user to reconnect.
- **Includes**: None
- **Notes/Issues**:
    * The Spotify integration needs to be thoroughly tested for reliability and performance.
    * The app should handle cases where the user's Spotify subscription is inactive.

6 Non-Functional Requirements

6.1 Performance Requirements

- The app should load within 2 seconds.
- The app should support real-time step tracking.
- The app shall store user data locally when offline and sync with Firebase when an internet connection is available.

6.2 Safety Requirements

- The app should only collect step count, fatigue data.
- The app shall include a pause tracking option.
- The app shall encourage user anonymity.
- The app shall not allow background step tracking without user consent.

6.3 Security Requirements

- The app shall use end-to-end encryption for storing and transmitting IMEI numbers in Firebase.
- The data collected would be stored for a year.
- The app will log users out if they are inactive for 30 minutes.

6.4 Software Quality Attributes

- The codebase shall be modular and well-documented, allowing for easy updates.
- The UI shall be simple and direct since most users have neurological problems.
- The app shall work on Android and iOS.

6.5 Other Requirements

- The app will comply with ISO 27002 security standards for cloud storage.

7 Appendix A: Glossary

- IMEI: Unique device identifier
- Firebase: Cloud backend service
- V1: Version 1
- V2: Version 2

7.1 Use Case Model

Figure 2 shows a general use case model for the project.

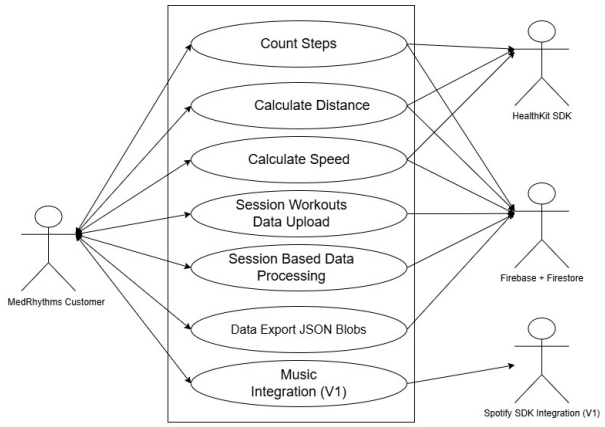

Figure 2: GENERAL USE CASE DIAGRAM FOR THE PROJECT

7.2 User Interface Sample

Figure 3 shows an initial wireframe of the login page, as well as the profile page and the session page.

Figure 3: User Interface

7.3 Sequence Diagram for Music Integration

Figure 4 shows a sequence diagram of music integration.

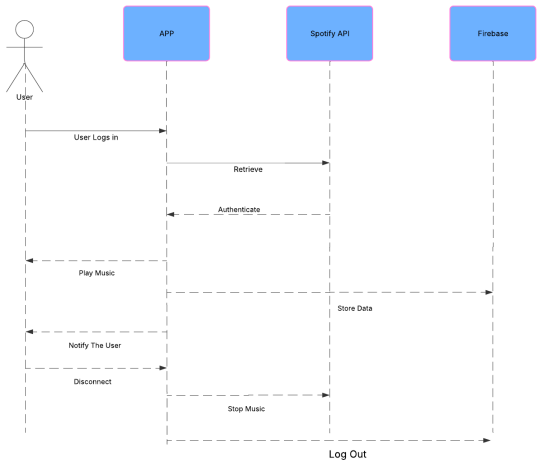

Figure 4: Sequence Diagram for music integration.