Software Requirements Specification
for
MEDRhythms Mobile App
Version 1.0

Prepared by

Group Name: Health App Development Team

Ineh Ifeanyi Robert (Primary Author)

Yiran Zhao

Yoga Srinivas Reddy Kasireddy

Instructor: Dr. Gary Cantrell

Course: Foundations of Software Engineering

Teaching Assistant: Sam Morris

Date:

February 16, 2025

Contents

1 Revision History

Version	Date	Description	Primary Author
1.0	February 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

The client for this project is MEDRhythms, and the stakeholders include the development team, project manager, testers, and end-users.

2.4 Intended Audience

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

2.5 Scope

The scope of the MedRhythms app allows users to monitor health metrics such as:

- Steps taken
- Speed
- Session Feedback

The app will integrate Apple Health for iOS devices and Google Fit for Android devices, with Samsung being the priority.

2.6 Definitions, Acronyms, and Abbreviations

- SRS – Software Requirements Specification
- 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

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

- V2 - Version 2

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.
- 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 7.0+ and iOS 13+ to ensure compatibility with Firebase as well as Google Fit and Apple HealthKit APIs.

3.6 User Classes and Characteristics

- Users with neurological conditions – To track steps taken
- Administrators – To maintain the system and monitor patients' data

4 Specific Requirements (Functional Requirements)

4.1 User Authentication

4.1.1 Description

The app authenticates users based on their IMEI number.

4.1.2 Priority

Essential

4.1.3 Stimulus

The user launches the app for the first time.

4.1.4 Response

The app retrieves the IMEI and logs in.

4.1.5 Functional Requirements

- The app shall have an option for the user to type in their IMEI number upon launch.
- The system then logs in the user after they input their IMEI.
- The app will store their IMEI records securely in Firebase Firestore.

4.2 Step Tracking

4.2.1 Description

The app tracks the users' steps, distance, and speed.

4.2.2 Priority

Essential

4.2.3 Stimulus

The user starts a session and begins walking.

4.2.4 Response

The app detects movement and updates the step counter.

4.2.5 Functional Requirements

- The app will make use of accelerometer and gyroscope sensors for tracking.
- The app will sync step count data with Firebase.
- The app will notify users when they have walked for 30 minutes.

4.3 Music Integration

4.3.1 Description

The app allows users to control music playback as well as play music.

4.3.2 Priority

Bonus

4.3.3 Stimulus

When the user starts working.

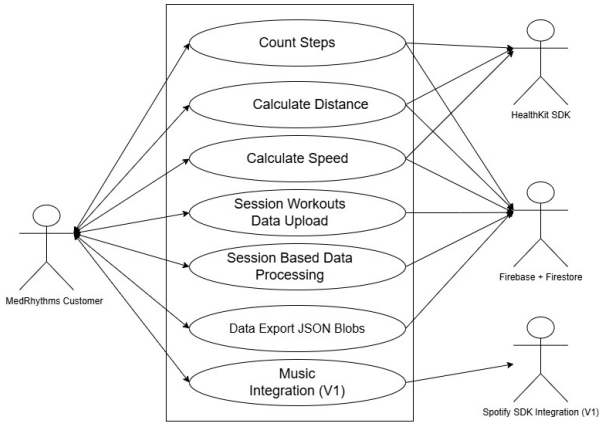
4.3.4 Response

It starts playing music to the rhythm of their steps.

4.3.5 Functional Requirements

- The app will have playback controls.
- The users can connect their Music account of choice.

4.4 Use Case Model



Use Case Diagram for MedRhythm Mobile App

4.5 Use Cases

4.5.1 UC1 - Use Case 1 - Count Steps

- Author: Yoga Srinivas Reddy Kasireddy
- Purpose: To track and count the number of steps a user takes during a session.
- Requirements Traceability:
  - I am a Registered User in the App.
  - I meet all the conditions for a walk-based workout.
  - The system must count the steps accurately.
  - Step data must be stored locally before being uploaded.
- Priority: High
- Preconditions:
  - The user has installed the app and granted necessary permissions.
- Postconditions:
  - Workout is complete, the data is processed into a format.
  - If the device is able to access the internet, data is uploaded to Firestore.

- Actors:
  - User – Walks and interacts with the app.
  - System – Detects steps and stores data.
- Extends: None
- 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.

4.5.2 UC2 - Use Case 2: Calculate Distance

- Author: Yoga Srinivas Reddy Kasireddy
- Purpose: To calculate the approximate distance traveled based on step count.
- Requirements Traceability:
  - I am a Registered User in the App.
  - I meet all the conditions for a walk-based workout.
  - The system must count the steps accurately.
  - The system should estimate distance based on step length.
- Priority: Medium
- Preconditions:
  - Step tracking is active.
  - We have permissions from the user to calculate the data.
- Postconditions:
  - Distance is calculated and stored for session tracking.
  - Uploads the data to Firestore.
- Actors:
  - User – Walks.
  - System – Converts distance and processes it.
- Extends: Count Steps
- Flow of Events:
  - Basic Flow:
    1. User starts walking.
    2. Step count is tracked.

3. System recognizes the distance the user walked.
4. Distance is displayed to the user and stored.

- Exceptions:
  - If the sensor is unavailable, we get a notification for the user to try again later.

- Includes: None
- Notes/Issues:
  - Needs optimization for different walking patterns (e.g., running, jogging).

4.5.3 UC3 - Use Case 3: Calculate Speed

- Author: Yoga Srinivas Reddy Kasireddy
- Purpose: To calculate the speed of movement based on step count and time.
- Requirements Traceability:
  - The system must calculate and update speed in real-time.
- Priority: Medium
- Preconditions:
  - I am a registered User of the app.
  - I meet all the pre-conditions for a walk.
  - Step, distance tracking is active and enabled by the user.
  - Time tracking is enabled.
- Postconditions:
  - Speed is calculated based on the session recorded and displayed to the user.
- Actors:
  - User – Moves at varying speeds.
  - System – Computes speed from step count and time or distance and time.
- Extends: Count Steps, Calculate Distance
- 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.
- Includes: None
- Notes/Issues:
  - May need testing to get accurate speed testing details.

4.5.4 UC4 - Use Case 4: Sessions, Workout, and Data Upload

- Author: Yoga Srinivas Reddy Kasireddy
- Purpose: To manage walking/running sessions and upload data when online.
- Requirements Traceability:
  - R5: The system must track step sessions.
  - R6: Data must be uploaded when the device is online.
- Priority: High
- Preconditions:
  - I am a registered User of the app.
  - I have completed my session.
  - Step, distance tracking is active and enabled by the user.
  - System has access to the Internet.
- Postconditions:
  - Session data is saved locally and uploaded to Firestore.
- Actors:
  - User – Starts and stops sessions.
  - System – Tracks session data and syncs when online.
- Extends: Count Steps, Calculate Distance
- Flow of Events:
  - Basic Flow:
    1. User starts a new session.
    2. Steps, distance, and speed are recorded.
    3. When the session ends, data is saved.
    4. If online, session data is uploaded to Firestore.
  - Alternative Flow:
    - If the user pauses a session, tracking is paused.
  - Exceptions:
    - If the device goes offline, data is stored locally and uploaded later.
- Notes/Issues:
  - Needs session summary statistics.

4.5.5 UC5 - Use Case 5: Session-Based Data Processing

- Author: Yoga Srinivas Reddy Kasireddy
- Purpose: To process step tracking data in session-based batches.
- Requirements Traceability:
  - I am a registered User of the app.
  - I have completed my session.
  - I have enabled all permissions for the app to analyze my session.
- Priority: High
- Preconditions:

  - A session is complete.
  - User has provided all the required permissions for analysis and storage of the data.
- Postconditions:
  - Session data is grouped and stored efficiently in the db in a safe format.
- Actors:
  - System – Processes session data.
- Extends: Sessions, Workout, and Data Upload
- Flow of Events:
  - Basic Flow:
    1. The system creates a new session record.
    2. Data is written in batch mode.
    3. Data integrity is verified.
  - Alternative Flow:
    * If the system is offline, we prioritize storing the data in local storage and then proceed to upload the data online.
  - Exceptions:
    * If batch write fails, the retry mechanism is triggered.
- Includes: None
- Notes/Issues:
  - Needs to handle large/multiple session sizes.

4.5.6 UC6 - Use Case 6: Data Export (JSON Blobs)

- Author: Yoga Srinivas Reddy Kasireddy
- Purpose: To allow users to export session data as JSON.
- Requirements Traceability:
  - All the data for the past sessions have been uploaded to the db.
  - System has access to this data.
  - The system must generate JSON exports.
- Priority: Medium
- Preconditions:
  - User has completed at least one session.
  - This data is successfully uploaded to the db.
- Postconditions:
  - A JSON file is generated and available for download.
  - It can be later uploaded to a different source for easy viewing and analysis.
- Actors:
  - User – Requests export.
  - System – Generates JSON file.
- Extends: None
- Flow of Events:

  - Basic Flow:
    1. User requests a data export.
    2. System fetches session data.
    3. JSON file is generated and made available.
  - Alternative Flow:
    * User selects a custom time range for export.
  - Exceptions:
    * If no data is available, an error message is shown.
- Includes: None
- Notes/Issues:
  - Needs validation for JSON format.

4.5.7 UC7 - Use Case 7: Music Integration (V1)

- Author: Yoga Srinivas Reddy Kasireddy
- Purpose: To integrate Spotify and play music based on user workouts or conditions.
- Requirements Traceability:
  - User has allowed Spotify integration.
  - User has the required playlist settings.
  - The app must connect to Spotify and play music.
- Priority: Medium
- Preconditions:
  - User has a Spotify account and grants access.
- Postconditions:
  - Music plays based on MedRhythms logic.
- Actors:
  - User – Starts music.
  - System – Connects to Spotify.
  - Spotify API – Provides music.
- Extends: None
- Flow of Events:
  - Basic Flow:
    1. User connects to Spotify.
    2. App retrieves song preferences.
    3. Music plays based on step count.
  - Alternative Flow:
    * User selects custom playlists.
  - Exceptions:
    * If Spotify API is down, music won't play.
- Includes: None
- Notes/Issues:
  - Needs testing for Spotify integration.

5 Non-Functional Requirements

5.1 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.

5.2 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.

5.3 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.

5.4 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 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 User Interface Sample

Figure 1 shows an initial wireframe of the login page, as well as the profile page and the session page.

Figure 1: User Interface

Appendix B: Group Log

- NIL