Software Requirements Specification

for

Fitness App

Version 1.0 approved

Prepared by Sam Zito and Adrian Zeni

Team: Cole, Amir, Troy, Landon, Adrian, Sam

Course: CSC 4710

Instructor: Dr. Williams

High Point University

2/16/2025

Table of Contents

1. Introduction

1.1 Purpose

This family fitness app is designed to encourage fitness among family members through data tracking, goal setting, and friendly competition. The app collects, stores, processes, and visualizes fitness data locally. The app also ensures privacy and security. It will support web and eventually mobile platforms so users can access easily from any location.

1.2 Intended Audience and Reading Suggestions

This document is intended for:
Developers who are implementing the app.
Project managers, Dr. Williams, that tracks the progress of the apps development.
End-users, families, who want to understand the app's specific features.
Testers that will work to ensure quality assurance.

1.3 Product Scope

This app will track several metrics including heart rate, weight, and water vs. muscle ratio. It will also allow family vs. family competition with fair heuristics. There will be a leaderboard for motivation and tracking. Features also include tunable personal goals and an AI component.

2. Overall Description

2.1 Product Perspective

This family fitness app is similar to other apps however, this is a new product that will integrate with smartwatches and fitness devices to help track user data and provide competitive tracking. This will allow our users to interact with their family members and their fitness goals easier, as well as friends. This app will also be interact-able on both mobile and web interfaces.

2.2 Product Functions

- Collects and stores fitness metrics
- Accessible on virtually any platform (web)
- Track and analyze long-term health trends
- Provides graphs and visualizations
- Supports family vs. family competition
- Uses an AI component
- Offers privacy controls on data sharing

2.3 User Classes and Characteristics

Family Members/Children

- Casual users who track fitness metrics
- Access to certain features limited
- May want privacy settings for some data

Admins (Family Leaders)

- Set up family competitions
- Adjust settings
- Modify accounts of family members, if necessary

2.4 Operating Environment

Supported Platforms:

- Web: Chrome, Firefox, Safari
- Mobile: Android, IOS

Operating Systems & Versions:

- IOS: Version 13 and later
- Android: Version 9 and later
- Windows: Windows 10 & 11
- MacOS: macOS 11 and later

2.5 Design and Implementation Constraints

- Should be lightweight to work on mobile and web
- Fast data processing
- Connection to outside software (i.e. Apple Health) possible, but may be limited
- Storing health data may require special security considerations

2.6 User Documentation

Components are listed:

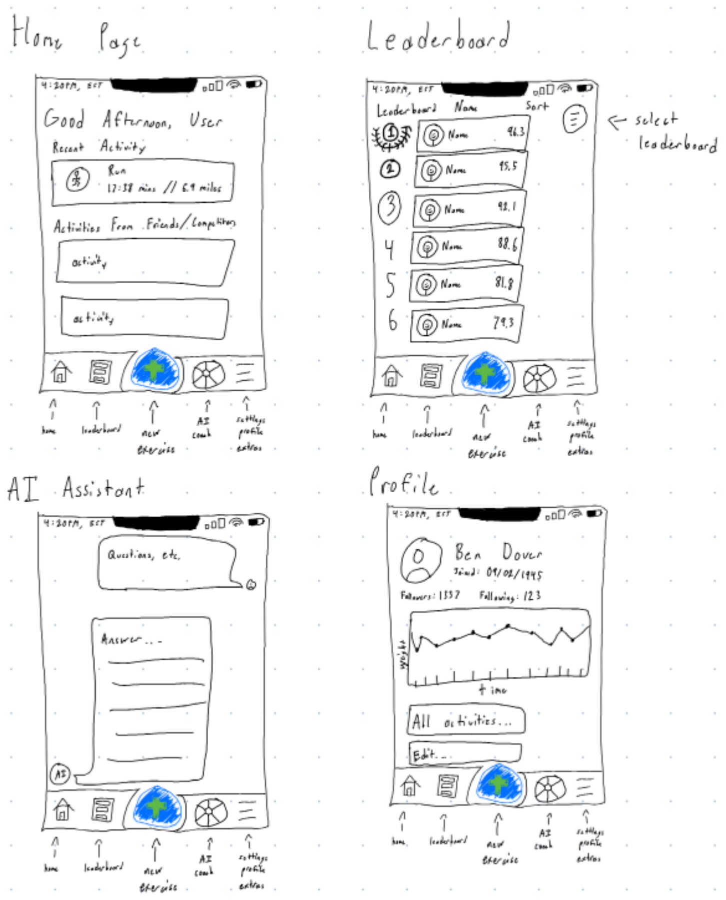
1. FAQs page on home website
2. Support via email/bug reporting page
3. Any other support via the community, either reddit page or discord

2.7 Assumptions and Dependencies

- Requires internet access
- Enough local storage for database
- Access to exercise recording (i.e. HR), if desired by user

3. External Interface Requirements

3.1 User Interfaces



- Home page with recent stats, leaderboard changes, etc. Includes a dashboard with graphs and metrics
- Leaderboard screen for competition
- Profile page for personal information
- Settings page for privacy controls
- Accessibility is available through the built-in screen reader that phones offer

3.2 Hardware Interfaces

Although we probably won't have time, a possible hardware interface is through a smart watch.

3.3 Software Interfaces

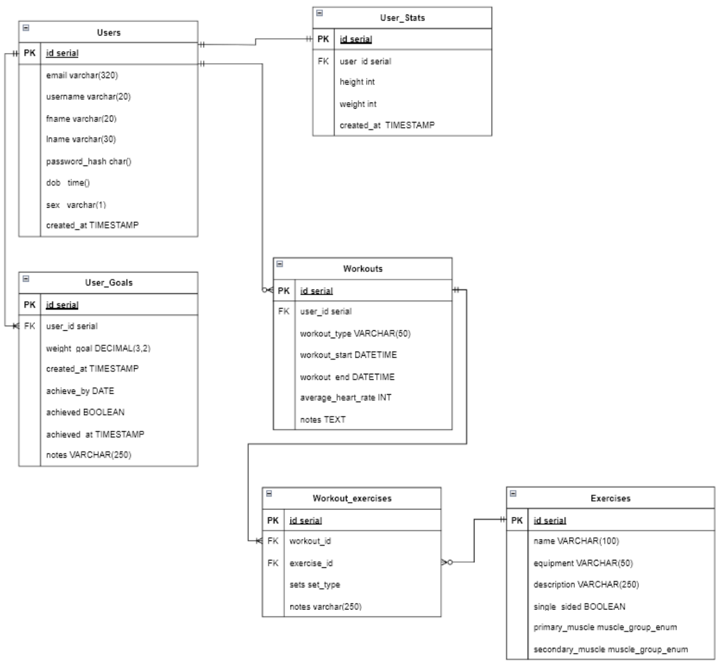
Database: Stores users fitness metrics
AI Module: Analyze data & suggest information
3rd Party API Integrations (Apple Health, etc.)

3.4 Communications Interfaces

Uses HTTPS for secure communication
Allows push notifications for fitness updates
Email notifications for changes in leaderboard positions, newsletters, etc.
Communicate with smart devices that use a common protocol, if possible

4. Data Design

4.1 Database ERD:



5. Software Development Approach

5.1 Development Methodologies Used

We chose a "Scrum-ban" approach as our development methodology, mainly because we have a diverse team with different strengths and weaknesses. This allows each team member to focus on their strengths, while reducing the time for a person to learn something they are not familiar with. Another reason we chose this process is because we have very limited time to complete this project, so our focus has to be on quick development with low overhead. Sprints from the scrum methodology allow us to focus on clear goals for a small amount of time. We will also be using a kanban board to keep us moving and it makes tasks transparent and easily visualized, as well as being able to know where all of our teammates are at a glance

5.2 Tools and Technology selected

- Frontend - React.JS
- Backend - Python (Flask) & KrakenD
- Database - PostgreSQL
- Testing - Docker
- Package Management & Deployment - npm and Expo

6. Test Plan

6.1 Overview

To make sure the fitness app functions correctly and meets the expectations of the customer, we will conduct the following testing strategies:

- Unit Testing: Each individual function will be tested separately, as necessary
- Integration Testing: Making sure frontend works with backend as expected
- System Testing: Validates the app with the functional and non-functional requirements
- Performance Testing: Test the response time and app stability
- Security Testing: Ensure user data protection
- User Acceptance Testing: Users test the app to make sure it meets their needs

6.2 Test Cases for Major Features

1. Ensure login works consistently
2. User inputs fitness data
3. Verify leaderboard updates after workout submission
4. Limit AI component to only fitness-related queries
5. Confirm that graphs show historical fitness data

6.3 User Acceptance Testing

- Define testing criteria
- Select test users
    - 10 - 15
- Test Execution
- Analyze Results
- Final Approval

7. Deployment Plan

7.1 Hosting Requirements

- requires:
  - Docker
  - Joecool
  - Expo

7.2 Steps to Deploy

For our instance of this project, we will be hosting the website and database on the joecool server. However, the final app would be best implemented on some sort of cloud system, using AWS or some other variation.

8. Other Nonfunctional Requirements

8.1 Performance Requirements

- App loads within 2 seconds
- Data sync should be under 30 seconds

8.2 Security Requirements

- User authentication required via login
- Users control data sharing settings

8.3 Software Quality Attributes

- Simple and intuitive UI (albeit, subjective)
- Available: 99.9% uptime
- Can handle a relatively large database

9. Future Considerations

9.1 Potential Improvements

- Wearable Device Integration: Google Fit, Apple Health, …
- Voice and Gesture Commands: Hands free workout logging via voice commands
- Optimize: Reduce response times for database queries
- End to End Encryption: Enhance security
- Two-Factor Authentication: Secure user accounts with email/SMS authentication
- Desktop Application: Provide cross-platform desktop app for fitness tracking

9.2 Known Limitations

- Limited smartwatch compatibility at launch
- Requires active internet connection for leaderboard updates

10. Conclusion

10.1 Summary

This software requirement specifications document lists the key requirements for the fitness app we are building. This app is designed to promote family fitness through data tracking, goal setting, and family competitions. It outlines the systems functionalities, architecture, testing, and security.

10.2 Next Steps

The next steps will include implementing core features, conducting tests, and deploying the app in incremental stages. Future improvements may expand on device integrations and enhanced security.