RUFit Design Document

**Authors:**

Naomi Wombough

Catherine Dougherty

Shiv Patel

Klaydon Balicanta

Erin Quigley

**Table of Contents**

[**1. Introduction**](#_yu70qnl3zr02) **3**

[1.1 Purpose](#_i46bqbpek29) 3

[1.2 Scope](#_nn9tejkm3aw1) 3

[1.3 Assumptions and Limitations](#_js1mkg6o07ad) 3

[**2 Design Specification**](#_jwq7hino2tj4) **3**

[2.1 Software Description](#_76vwf1jzwpik) 3

[2.2 Firebase Service](#_83o1ak1s9wsb) 4

[2.3 Database Design](#_gfeq44yld98f) 6

[2.4 Design Prototype](#_eznh83d8eej9) 7

[**3 Implementation Plan**](#_ytjybxkm272f) **8**

[3.1 Tasks](#_7jw4ar4bcecf) 8

[3.2 Roles and Responsibilities](#_beibb0bzcnta) 9

# 

# 

# 1. Introduction

This Application Design Document has been created to outline the design for the new RUFit Android application, where RUFit is short for “Rowan University Fit”. The RUFit app is intended to be a unique alternative to other similar applications that already exist for Android devices.

## 1.1 Purpose

The purpose of this document is to outline the software design of the RUfit application described in the Specification Document. The RUfit application is a run tracking application targeted to competitive runners. The application will run on the Android operation system.

## 1.2 Scope

This document aims to be a guide for the developers to implement the prospective RUfit application. This document contains details about the application’s usage and implementation.

## 1.3 Assumptions and Limitations

The application will track user's workout using GPS from the devices and allow them to analyze and share their workouts to improve their health habits. The RUfit app will only track a user’s running/walking. And the application will be supported by devices running Android OS.

# 2 Design Specification

## 2.1 Software Description

RUFit App will be developed in JAVA programing language using Android SDK, Google Firebase technologies and Facebook SDK. The App will be tracking locations using devices GPS to compute a user’s run.

Android SDK is a set of development tools used to create beautiful Android apps.

Firebase Technologies will be used for Authentication and Database purpose in the App.

Facebook SDK will be used to share the app and users contents to Facebook.

## 2.2 Firebase Service

The RESTful API is the program interface which uses HTTP requests to describe the architecture and communications between client and server. Listed are the endpoints for each client action and how it will interact with the server.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Endpoint | Purpose | Input | Output | Implementation |
| POST /login/ | Find username and password for login | [username]  [password] | Success : boolean  userID : String | -User inputs username and password to be authenticated by Firebase |
| POST /signup/ | Create new account | [email]  [username]  [password]  [phone]  [date] | Success : boolean | - User will enter criteria for new user to be added |
| POST /user/runs/add/ | Add new run to log | [userID]  [date]  [name]  [distance]  [time]  [shoes]  [feel]  [type]  [notes] | Success : boolean | -Criteria for each run will be added under each user’s branch within Firebase |
| GET /user/runs/ | View runs on given date | [userID]  [date] | Array of :  Date : Date  Name : String  Distance : double  Time : int  Shoes : enum  Type : enum  Feel : enum  Notes : String  Success : boolean | -Given date, all run data matching that date is returned |
| PUT /user/runs/ | Edit log entry | [userID]  [date]  [name]  [distance]  [time]  [shoes]  [feel]  [type]  [notes] | Success : boolean | -Using previous endpoint to retrieve run information, information can be edited and updated to Forebase. |
| GET /user/records/ | Retrieve user’s best records | [userID] | longestRun : double  fastestMile : String  longestStreak : int  Success : boolean | -Get record data from user’s branch to view records, or compare most recent log to record data in case it needs updating |
| PUT /user/records/ | Update user’s best records | [userID]  [longestRun]  [fastestMile]  [longestStreak] | Success : boolean | -After a run is logged, records data is compared to current run for updating |
| GET /user/shoes/ | Retrieve user’s shoes | [userID] | Array of :  shoeName : String  mileage : double  Success : boolean | -When user wants to view shoe archive and mileage on shoe, all shoes and data is returned |
| PUT /user/shoes/ | Update miles ran in shoes | [userID]  [shoeName]  [milesAdded] | Success : boolean | -After each run, the shoe the user associates with the run is retrieved and the number of miles from the current run is added to shoe’s total mileage |
| POST /user/shoes/ | Add shoe to archive | [userID]  [shoeName] | Success : boolean | -When user wants to submit new shoe, they just enter name of the shoe |
| GET /user/info/ | Retrieve personal info | [userID] | Gender : String  Height : int  Weight : int  Age : String  Email : String  Phone : String  Success : boolean | -Personal information for user is retrieved |
| PUT /user/info/ | Update personal info | [userID]  [gender]  [height]  [weight]  [age]  [email] | Success : boolean | -After personal information is retrieved, user has ability to update fields. |
| GET /user/goals/ | Retrieve goals | [userID] | milesPerWeek : double  runsPerWeek : int  daysUntilRace : int  Success : boolean | -When user wants to check on their goals, goal data will be retrieved. All goal fields are initialized to 0 or null, user much opt in and select goals to follow |
| PUT /user/goals/ | Update user’s goals | [userID]  [milesPerWeek]  [runsPerWeek]  [daysUntilRace] | Success : boolean | -After goal data is retrieved user has ability to customize goals to reach. |

## 

## 2.3 Database Design

The following are representations of the tables containing what entities and their accompanying attributes will be associated with.

**User Table**

|  |  |
| --- | --- |
| **Entity** | **Attribute, NULL Value (If NULL, Defaults applied)** |
| firstName | Character, NOT NULL |
| lastName | Character, NOT NULL |
| password | Character, NOT NULL |
| email | Character, NOT NULL |
| phoneNumber | Integer, NULL, default: 000-000-0000 |
| dateAccountCreated | Date, NULL, default: “today’s” date (that is, present date) |
| username (PK) | Character, NOT NULL |
| recordPace | runID (FK), NULL, default: 0 |
| recordDistance | runID (FK), NULL, default: 0 |
| recordTime | runID (FK), NULL, default: 0 |
| milesPerWeek | runID (FK), NULL, default: 0 |
| runsPerWeek | Integer, NULL, default: 0 |
| daysUntilRace | Integer, NULL, default: -1 |

**Run Table**

|  |  |
| --- | --- |
| **Entity** | **Attribute, NULL Value (If NULL, Defaults applied)** |
| userID (FK) | Username-character, NOT NULL |
| shoeID (FK) | shoeID-integer, NOT NULL |
| runID (PK) | Integer, NOT NULL |
| pace | Float, NOT NULL |
| distance | Float, NOT NULL |
| time | Integer, NOT NULL |
| feel | character/enum, NULL, default: “Normal” |
| type | character/enum, NULL, default: “Daily Run” |
| notes | character , NULL, default: “<none>” |
| date | Date, NOT NULL |

**Shoe Table**

|  |  |
| --- | --- |
| **Entity** | **Attribute, NULL Value (If NULL, Defaults applied)** |
| shoeID (PK) | Integer, NOT NULL |
| userID (FK) | Username-character, NOT NULL |
| shoeName | Character, NULL, default: “Shoe\_x” where x incrementally increases from 1 for each new shoe added to the table |
| mileage | Integer, NULL, default: 0 |

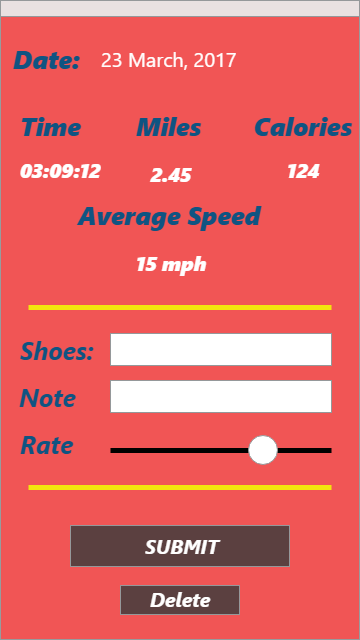
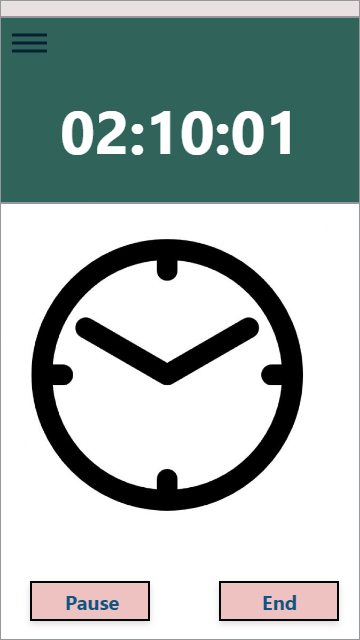
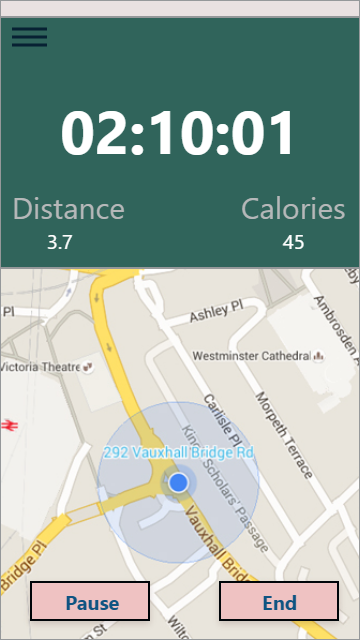
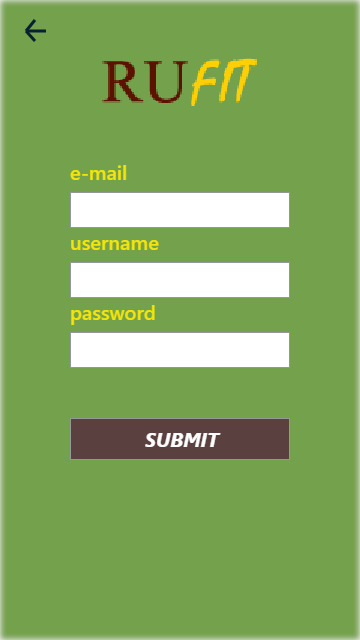
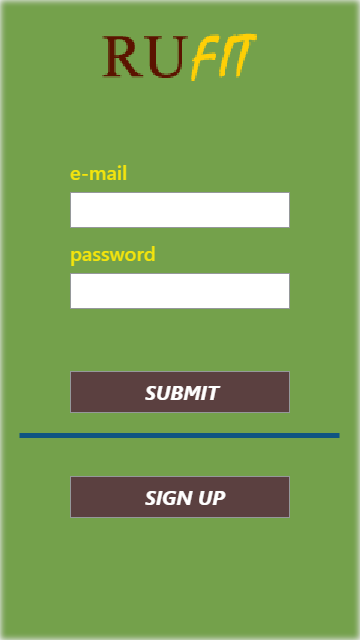
**Goal Table**

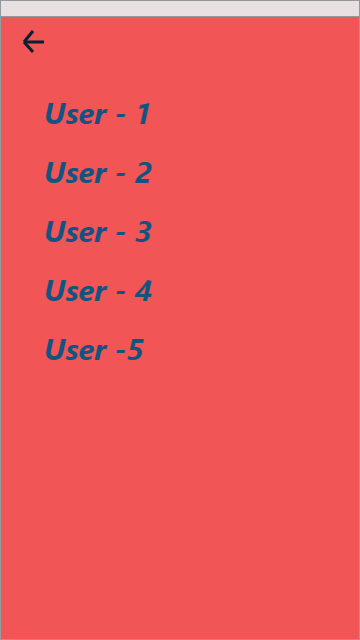
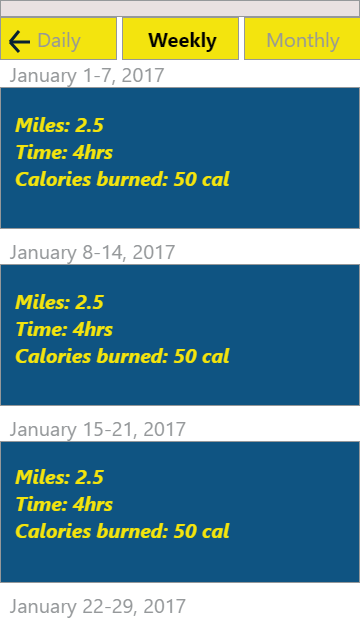
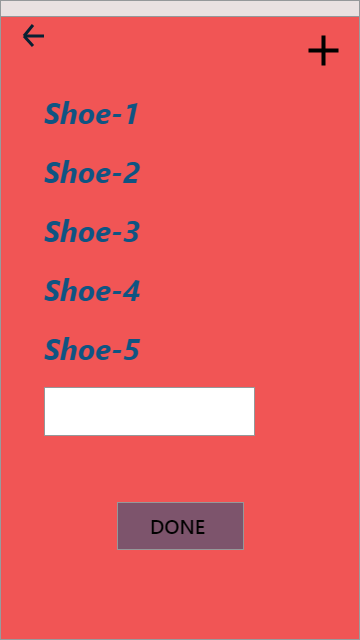
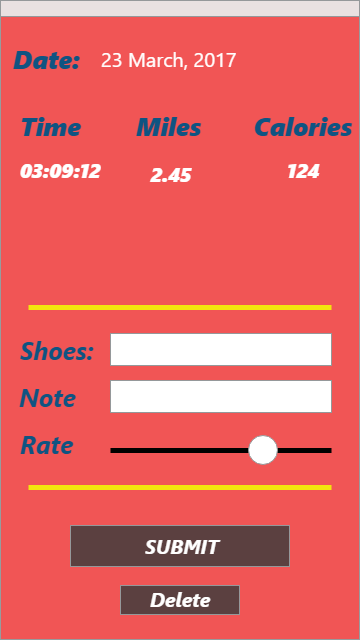
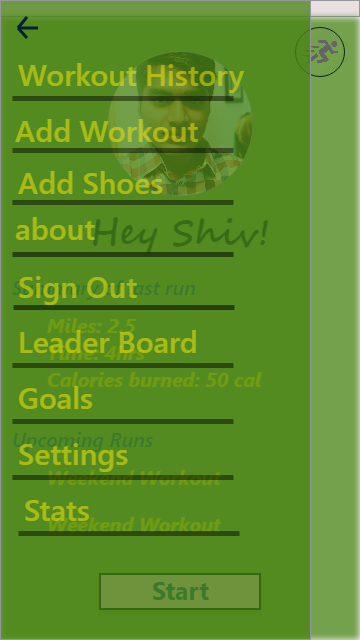
|  |  |
| --- | --- |
| **Entity** | **Attribute, NULL Value (If NULL, Defaults applied)** |
| userID (FK) | Username - character, NOT NULL |
| weeklyMileage | Integer, NULL, default: 0 |
| runsPerWeek | Integer, NULL, default: 0 |

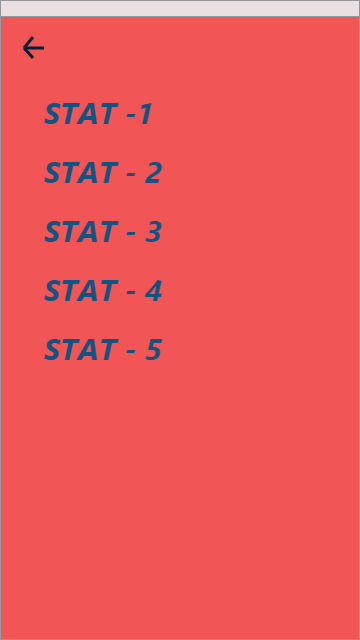
## 

## 2.4 Design Prototype

Working prototype: https://xd.adobe.com/view/ef225ff4-1fa7-48e0-ba68-5fe8bba83b29







# 3 Implementation Plan

Below is a table showing the tasks, roles, and responsibilities of each team member for the first two weeks of the implementation plan. Assignment of these tasks are not fully committed as all team members will work together to ensure all tasks are completed within a timely manner. The roles and responsibilities of each team member will interchange commonly to allow each member to approve, input, and understand the project fully.

## 3.1 Roles, Responsibilities, and Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| ***Members*** | ***Roles*** | ***Responsibilities*** | ***Tasks*** |
| Naomi Wombough | Development | Creates a fully functional product adhering to the Specifications and Design Documents | POST /user/runs/add/ endpoint  GET /user/runs/ endpoint  PUT /user/runs/ endpoint |
| Klaydon Balicanta | Database Design/  Development | Assign proper architecture of tables with entity and attribute association | Set up database architecture |
| Erin Quigley | Development | Creates a fully functional product adhering to the Specifications and Design Documents | POST /login/ endpoint  POST /signup/ endpoint |
| Catherine Dougherty | Database Design/  Development | Assign proper architecture of tables with entity and attribute association | Set up database architecture |
| Shiv Patel | Firebase Design/ GUI Development | Managing Firebase database and authentication. | Handle user authentication  Create GUI prototype |

## 