

## Team

Jesse Bernoudy - Backend development

Bob McHenry - UI development

## Abstract

Our app will take advantage of the FitBit REST api to provide an easy to read, at-a-glance visualization of the users daily progress. The visualization will be a character sprite moving along a themed background, turning a user's FitBit progress into an idle-style game. The app will require the use of FitBit's REST api as well as the RetroFit Android api to adapt the REST calls to the Android ecosystem. The app will also require a function to launch the FitBit app, in the case the user manually syncs their FitBit data. This app will be targeted at game enthusiasts who are looking for something extra to add a little fun into their personal health care as well as anyone else looking for a more interesting display of their daily progress.

## Use Case:

As a lazy gamer, I don't care if my FitBit app shows me numbers as much as I care that my game character advances throughout a level, making me more likely to move around a bit to make progress in the game.

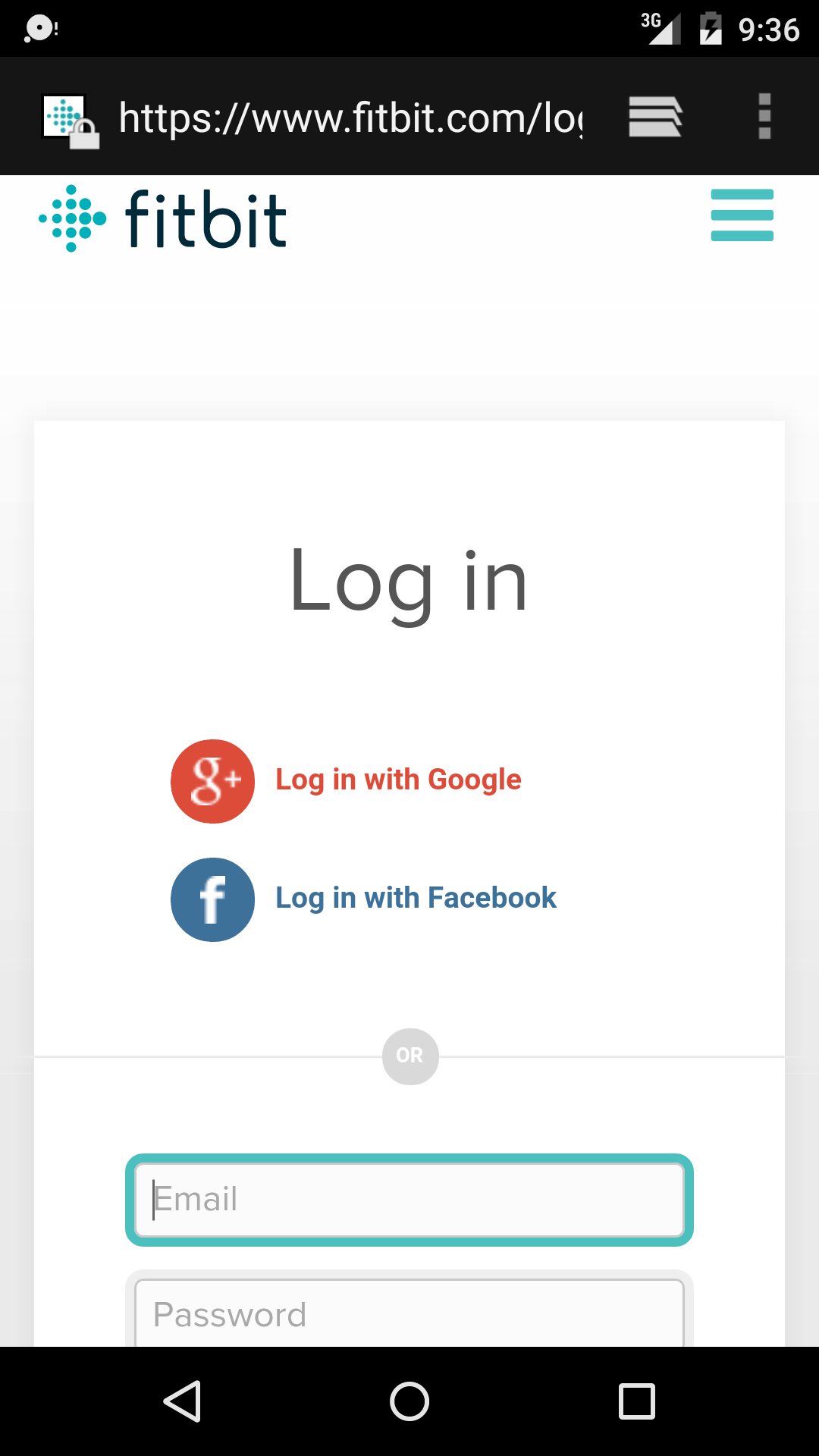
## 

## 

## Screens

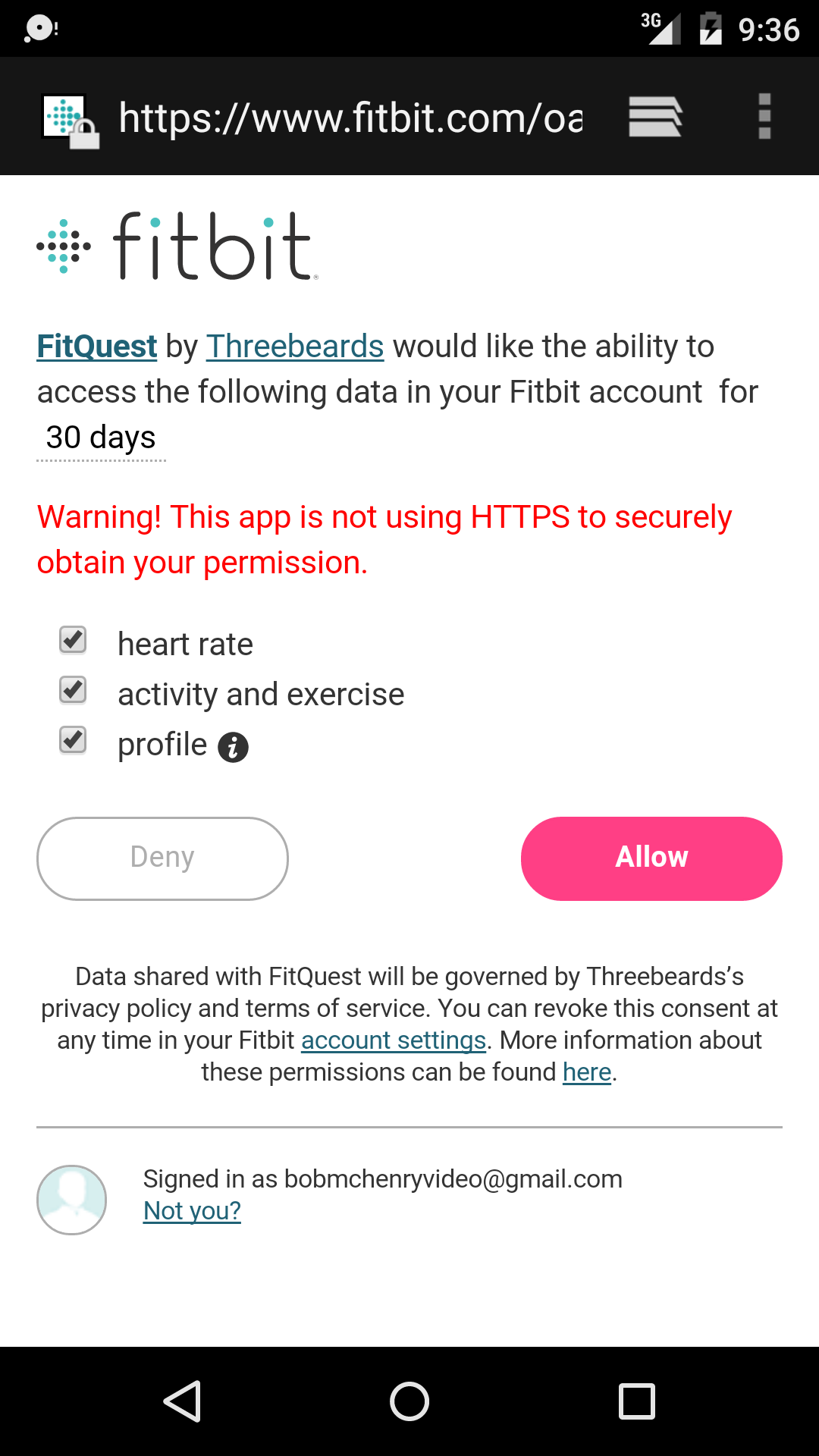
### Login:

When first logging into the app, the user is redirected to the fitbit website to acquire OAUTH2 authorization for FitQuest.



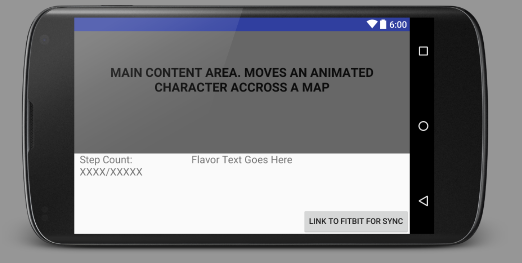
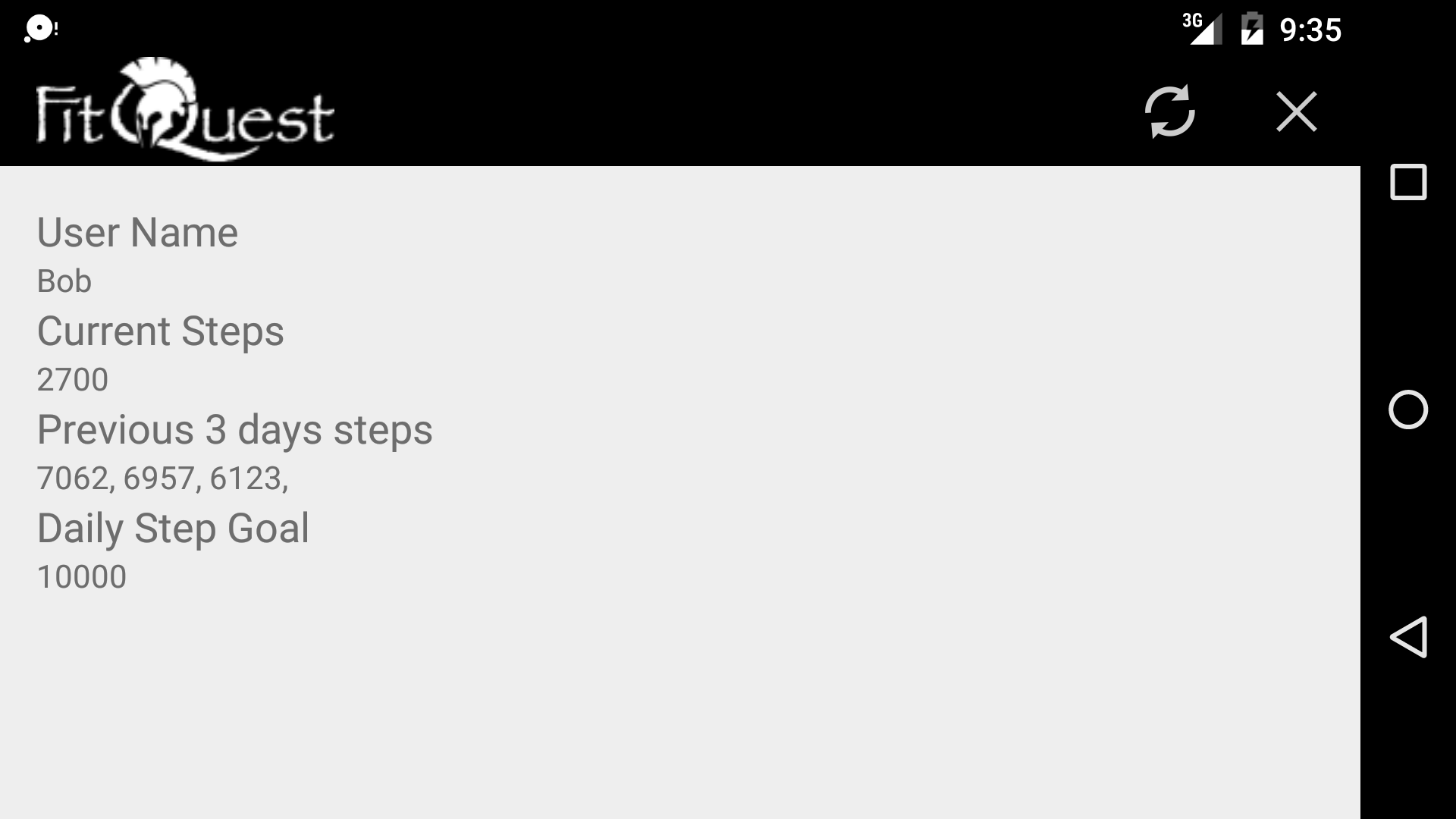
### Authorization:

After logging into fitbit, the user is presented with the authorization screen which prompts them to allow FitQuest to have access to limited data



### Main App screen:

The main app screen will show the user's daily step progress. Tapping the map or clicking the refresh () button will update the character and animate the progress across the map. The user can logout at any time by clicking the logout () button



## 3rd party APIs Used

### FitBit (<https://dev.fitbit.com/>)

Used to retrieve users daily step count, name, step count history, and more from their FitBit account.

### Retrofit (<http://square.github.io/retrofit/>)

Used to make authenticated REST calls to the FitBit api, relies on OKHttp3 behind the scenes

### Jsonschema2pojo (<http://www.jsonschema2pojo.org/>)

Used for auto generation of classes from JSON schema to deserialize REST responses.

## Architecture

The application is split into two main categories, REST/Oauth layer, and the UI layer. The REST Layer use Retrofit to abstract the calls to the FitBit api, and handle user authentication. It is made up of the following classes

### Backend/Data Layer

#### ServiceGenerator.java

Wrapper around the Retrofit Api to handle user authentication

#### FitBitServiceGenerator.java

Wrapper for FitBit specific URL’s to use with the ServiceGenerator

#### FitBitApiService.java

Interfaces for the REST calls FitQuest makes to FitBit

##### FitBitUser.java

Class that abstracts and expose the logged in users information that the app is interested in. Network calls are made on their own threads and the calling class is updated through the implementation of the FitBitUserListener interface to ensure the UI thread is not blocked.

##### ActiviiesStep.java, Activity.java, CustomHeartRateZone.java, Distance.java, Fetures.java, Goals.java, StepHistory.java, Summary.java, User.java, UserActivity.java, UserProfile.java

Autogenerated classes from <http://www.jsonschema2pojo.org/>

### UI Layer

The UI Layer handles launching out to authorize the user, saving user data in SharedPreferences and the animations. It consists of the following classes

#### HomeActivity.java

Only activity in the application, it contains the FitBitUser instances and calls proper refresh methods during the application lifecycle. It also handles preserving user data in SharedPreferences

#### WalkingView.java

This class is an extended view that contains an animate function to increase the characters distance across the screen.