Logisteps Mobile App V1

# Overview

This document describes the high-level documentation for using the Logisteps mobile application. The mobile device serves as a middle ground between the physical insole and the web server that holds all the data. The mobile app also serves as a quick glance into the users personal data being collected.

# Current Version

This is the first version of the Logisteps mobile app. Updates to the app, including changes and new features, will be documented in the upcoming versions of the design documentation. The latest version can be viewed at this location:

<https://github.com/SeniorDesignTeamOmicron/Documentation/blob/master/Project%20Design/mobile%20application/current/mobile%20app%20v1.docx>

# Schema

The app communicates in two different ways. One is to the server where it uses OkHTTP. The other is to the insoles where it uses Bluetooth. The app is designed to work on android API 26 or above. This API allows a few of the older mobile devices to use this technology. All step data is posted to the web server in the format shown in the Rest Framework documentation here: <https://github.com/SeniorDesignTeamOmicron/Documentation/blob/master/Project%20Design/web%20application/current/current%20version.txt>

The data format received from the insoles is documented here:

Put Hyperlink to insole design documentation here.

# Users

The server API holds all the user data. The mobile app, however, needs a way to access it to post steps and to show the basic information. This is done through signing in to the app or creating an account. The app then verifies with the server that it is in fact a user or the user has been created. The responses from the server decide what happens next in the app.

## Basic Authentication

The authentication code is created with base 64 encoding. For all posts and requests to the server the user data must be encoded. When the user attempts to sign in, the mobile app connects to the server, and sends the encoded user sign in. It then waits for a response. As shown in the Rest Framework documentation, passing valid credentials returns a response of “200 OK”, then it sends the user data. The response is used to set the step goal and ensure the data is correct in the app.

The encoding is done with the username and password. It connects them as “<username>:<password>”. It then uses that string to create a code. That code is attached to the authentication string as: “Basic <encoded>==”

### Parameters

|  |  |  |
| --- | --- | --- |
| Username | String | Users unique name that will be associated with their data |
| Password | String | Users password connected with the above username in order to access their data |

## Create a user

Creating a user is done in the log in screen. A pop up comes up when the create user hyperlink is pressed. With the pop-up comes more options as shown below. All the fields must be filled in before the user can be created. If one is not filled in, the app shows a toast that requests all fields filled in. Once the fields are filled in, the app passes the data to the server in a OkHTTP post to the /api/user page, then waits for the response. A response of “201 Created” is ideal, however, upon failure the server will return with “400 Bad Request”. If that is to happen, the error will be presented to the user and the user can then correct the error. For example, if the user is already created and the user may have forgotten, they can now return and sign in.

### Parameters

|  |  |  |
| --- | --- | --- |
| Username | String | Users unique name that will be associated with their data |
| Password | String | Users password connected with the above username in order to access their data |
| email | String | Valid email that user can be contacted at |
| first\_name | String | First name of the user |
| last\_name | String | Last name of the user |
| password | String | Plain text password for user. Will be encrypted prior to being stored |
| right\_shoe.size | Float | Size of the user’s right foot. Precision should be limited to 1 decimal. Valid sizes are between 4 and 16. |
| left\_shoe.size | Float | Size of the user’s left foot. Precision should be limited to 1 decimal. Valid sizes are between 4 and 16. |
| height | Integer | Height of the user in inches |
| Weight | Integer | Weight of the user in pounds |
| step\_goal | Integer | User’s daily step goal. Must be greater than or equal to 0. |

## Failed Authentication

Upon failure to authenticate with the server, the server responds with “400 Bad Request”. It then shows an error message for what is bad about the request. The message that is passed will be presented to the user when it is user error, so the user can correct based on the error. If it is application error, the app will correct itself and try again. It will time out after some attempts and request the user to try again.

## Change Account

When in the main screen, after logging in, the user has an option to change their account. The app stores recent valid credentials in its internal storage. When the user selects change account, all the usernames available will pop up. When the user selects one, the app will verify that it is a valid credential, for redundancy, and get the new data from that user by a get user request to the server. It will then update the interface with its new parameters and data.

# Steps

Steps are the main tracking of the LogiSteps design. Since the mobile app acts as a midpoint, it will see all the steps that come from the insoles and pass them to the server. The step data from the insoles will have the pressure data and the time. The app will attach on the user data and the location before sending it to the server.

## Posting Steps

Steps will be posted through an OkHTTP post. The data that is posted is shown below and is collected from internal mobile services and the actual LogiSteps insole. A return value of “201 Created” confirms that the data was posted, and the server has accepted. This also requires authorization to be passed in order to know where to store the data.

### Parameters

|  |  |  |
| --- | --- | --- |
| datetime | String | String representation of an ISO-8601 datetime object with timezone offset. |
| sensor\_reading.location | Char | Character representing the location of the sensor on the shoe. “T” for top, “B” for bottom. |
| sensor\_reading.pressure | Float | Pressure recorded for the step |
| sensor\_reading.shoe | String | Indication of which shoe the step was taken. “right” for the right shoe, “left” for the left shoe. |
| right\_shoe.size | Float | Size of the user’s right foot. Precision should be limited to 1 decimal. Valid sizes are between 4 and 16. |
| location.latitude | Float | Latitude of user’s location |
| location.longitude | Float | Longitude of user’s location |

## Get Step Summary

In order to update the user’s data shown in the main screen, the app must poll the server for the day data. It only shows the current day so that is all that is needed. The app uses a get request to the “/api/steps/summary/?date=mm-dd-yyyy” page. This requires authorization as well it returns all the data needed for the day. The summary will be polled every two minutes to keep an up to data display on the app. The total steps and the steps per hour are used to calculate a projected step count for the day to display to the user. That is the only thing that the app shows that is not in the server.

### Returned

|  |  |  |
| --- | --- | --- |
| steps | Int | The total steps for the day |
| goal | Int | The users step goal |
| percent | Float | The percent of steps the user has done |
| least\_active.hour | Int | The hour after midnight that the user was the least active |
| least\_active.steps | Int | The number of steps that were taken in the least active hour |
| most\_active.hour | Int | The hour after midnight that the most steps were taken |
| most\_active.steps | Int | The number of steps in the most active hour |
| inactive\_time.hours | Int | The number of hours that there was no activity |
| inactive\_time.minutes | Int | The number of minutes there was no activity |
| steps\_per\_hour | Float | The number of steps in an hour |

# Connection

There area three connections managed with the app. The first two are to the two insoles and the third is to the server. Each is managed inside the app. They are each managed in different locations as well.

## Server

The server IP address is shown in the bottom of the log in screen. The user can change it if the server has moved. This is only the base IP address and not the whole URL needed. The user can also change the URL in the main screen of the app by clicking on the circular button at the top. If this button is green, the server is connected. If it is red, the server is in error. By clicking on the button and attempting to connect, the user can see the error the server is giving. With it, it can find out what is happening and correct the error.

## Insole

The insole is connected through Bluetooth. The connection is managed with an object for each foot. The user has the option to connect simply by clicking on which foot they are connecting to and then by clicking connect. The name of the Bluetooth device is used to decide which foot is sending data, so the insole cannot have a null name. it must have a name that is unique as well. There will be an error if the Bluetooth is not connected or is connected incorrectly.