Software Design Description

Version 1.0

November 3, 2019

Fabio Calero, Sudara Ranasinghe, Jihee Son, Kwangmin Kim

**Table of Contents**

[**1.0. Introduction** 3](#_Toc23938103)

[**1.1. Purpose** 3](#_Toc23938104)

[**1.2. Scope** 3](#_Toc23938107)

[**1.3. Overview of document** 3](#_Toc23938111)

[**2.0 Architecture Design 5**](#_Toc23938118)

[2.1 Web System Architecture 5](#_Toc23938119)

[**2.1.1. Login** 6](#_Toc23938120)

[**2.1.2 SignUp** 7](#_Toc23938121)

[**2.1.3 Main page** 9](#_Toc23938122)

[**2.1.4 Database** 12](#_Toc23938172)

[**3.0 Data Structure Design 13**](#_Toc23938173)

[**4.0 Use case realizations 15**](#_Toc23938174)

[4.1 Use Cases 15](#_Toc23938175)

[4.2 Use Case Realizations 17](#_Toc23938178)

[**4.2.1 Use Case: User log-in and log-out** 17](#_Toc23938179)

[**4.2.2 Use Case: Zoom in on a place and zoom out** 18](#_Toc23938180)

[**5.0 Interface Design 19**](#_Toc23938181)

**6.0 Help System Design …………………………………………………………………………………………………………………… 25**

**1.0. Introduction**

**1.1. Purpose**

This Software Design Document provides the design details of the CSE316 TAK group traveling website (Bucky List) (<https://www.buckylist.com>).

The expected audience is all men and women from ages 0 to 65 interested in planning their vacation travels online. The Bucky Website will serve as a hub for users to explore places, read and write reviews of certain destinations, as well as allowing the users to create itineraries by adding places they would like to visit to a bucket list.

**1.2. Scope**

This document contains a partial description of the architecture and design of Bucky List. Some of the features implemented into Bucky List will be incorporated into this design document at a later date.

Since the project follows a Scrum development strategy, this document is meant to serve as the design document for the initial sprints of the development process.

The basic architecture is a webserver from a client server paradigm. The basic pages will be made dynamically using JavaScript (react). By using react we limit our web server to only deal with the database, putting the workload of the processing logic on the client side.

**1.3. Overview of document**

The remaining chapters and their contents are briefly explained below.

Section 2 has the Architectural Design of the application. This section contains the design entities that interact together to perform all the functions included in the system. Each of these entities has a name, description, and a lower-level set of design operations that collaborate to perform its services.

Section 3 contains the Data Structure Design where the data structures used in the system are specified to conform with the required functionality of the system.

Section 4 contains the Use Case Realizations.

Section 5 discusses the User Interface Design. Possible implementations of the interface design are shown based on prototypes of the web application. The User Interface Design focusses on ease of use and simplicity.

Section 6 contains information on the help system.

# 2.0 Architecture Design

## 2.1 Web System Architecture

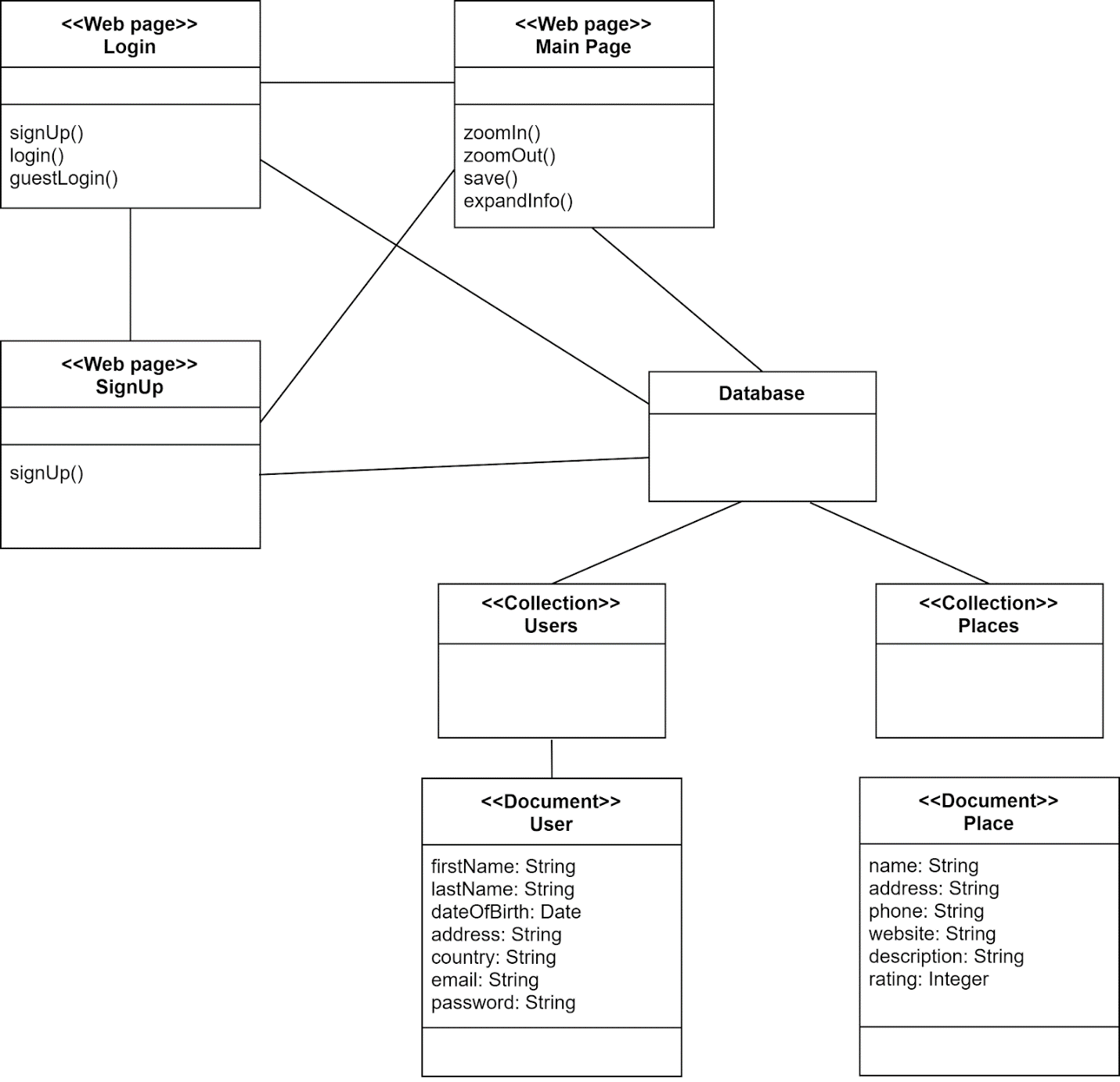


Figure 2.1.1 Web System Architecture

### **2.1.1. Login**

Name: Login

Type: Web page

Description: This is the initial page presented to the user upon connection with the website. There are 3 buttons. The buttons have the following functions:

1. *Login to the website* after the user has entered the user credentials on the Login page. The user credentials are:
   1. The user’s username or email address
   2. The user’s password
2. *Sign up* with the application, i.e., create a user account in the application. When the user clicks the sign-up button the user will be redirected to the SignUp page.
3. *Log-in as a guest*. If the user clicks the login-as-guest button the user will be redirected to the *Main page*, without the need to authenticate the user.

Attributes: None

Operations:

Name: login()

Arguments: userName, password

Returns: No return value

Pre-condition: On the login page. The user is not authenticated.

Post-condition: On the Main page. The user is authenticated by the application.

Exceptions: Invalid user-name and/or password

Flow of Events:

1. The user is presented with the Login Page.

2. The user enters the user credentials (described above).

3. The user clicks on the login button.

4. The user credentials are authenticated.

5. The user is directed to the Main page.

Name: signUp()

Arguments: None

Returns: No return value

Pre-condition: On the login page. The user is not authenticated.

Post-condition: On the SignUp page. The user is not authenticated.

Exceptions: None

Flow of Events:

1. The user is presented with the Login Page.

2. The user clicks on the sign-up button.

3. The user is directed to the SignUp page.

Name: guestLogin()

Arguments: None

Returns: No return value

Pre-condition: On the login page. The user is not authenticated.

Post-condition: On the Main page. The user is not authenticated.

Exceptions: None

Flow of Events:

1. The user is presented with the Login Page.

2. The user clicks on the login-as-guest button.

3. The user is directed to the Main page.

### **2.1.2 SignUp**

Name: SignUp

Type: Web page

Description: The SignUp page is presented to the user upon clicking on the sign-up button on the login-page. The signUp page contains a form where the user can enter the requested information. The following fields in the form are required:

1. First Name
2. Last Name
3. Date of Birth
4. Email
5. Password

The following fields are optional:

1. Country of residence
2. Address

The email is verified by sending a verification link to the email address. The password is checked for adequate length and strength, and the user is also required to reenter the password.

Attributes: None

Operations:

Name: signUp()

Arguments: Form object

Returns: No return value

Pre-condition: On the SignUp page. The user is not registered with the application.

Post-condition: On the Main page. The user is registered with the application.

Exceptions: Invalid or missing fields in the form.

Flow of Events:

1. The user is presented with a sign-up form.

2. The user enters the requested information.

3. The user clicks on the sign-up button.

4. The fields in the form are validated.

5. The user is registered with the application.

6. The user is directed to the Main page.

### **2.1.3 Main page**

### Name: Main

### Type: Web page

### Attributes: None

### Description: This is the application’s main page which is presented to the user upon successful login. While in the main page, the user is presented with an interface consisting of an account panel on the left and a world map on the center of the page. The user is able to do four main operations on the map: zoomIn, zoomOut, save, and expandInfo. Using these operations the user can explore the world map on the center of the page and also bring about additional panels to appear containing more information about the places in the map. The four allowed operations on the map are detailed below:

### 

### Operations:

### Name: zoomIn()

### Arguments: None

### Returns: No return value

### Pre-condition: On the Main page

### Post-condition: On the Main page. The map view is zoomed around the pointer position.

### Exceptions: None

### Flow of Events:

### 1. The user is presented with the Main Page.

### 2. The user hovers over a place of interest.

### 3. The user uses mouse scroll wheel up or double click on the place of interest.

### 4. The map zooms in about the pointer position.

### 

### Name: zoomOut()

### Arguments: None

### Returns: No return value

### Pre-condition: On the Main page. The user has previously zoomed in.

### Post-condition: On the Main page. The map view is zoomed out around the pointer position.

### Exceptions: None

### Flow of Events:

### 1. The user is on the Main Page.

### 2. The user hovers over a place of interest and zooms in.

### 3. The user uses mouse scroll wheel down or right click about the place of interest.

### 4. The map zooms out about the pointer position.

### Name: save()

### Arguments: string place

### Returns: true upon successful save.

### Pre-condition: On the main page. The user has zoomed into a section of the map and clicked on a place to select it.

### Post-condition: On the Main page. The selected place has been added to the Bucky List.

### Exceptions: None

### Flow of Events:

### 1. The user is presented with the Main Page.

### 2. The user zooms in on the map.

### 3. The user clicks on a place to select it.

### 4. The user clicks on Add to Bucky button.

### 5. The place is added to the Bucky list.

### 

### Name: expandInfo()

### Arguments: None

### Returns: No return value

### Pre-condition: The user is on the main page and must have selected a destination. The account panel must be displaying with collapsed description, reviews, and flight information sections.

### Post-condition: The user is on the main page with a destination selected. The account panel must be displaying with expanded description, reviews, or flight information section according to the user’s selection.

### Exceptions: None

### Flow of Events:

### 1. The user is presented with the Main Page.

### 2. The user clicks on a collapsed section on the left panel.

### 3. The user is presented with the contents of the previously collapsed section.

### **2.1.4 Database**

Name: Database

Type: Database

Description: The database is a no-SQL database hosted in Google Firebase as a Cloud Firestone solution. The structure of the database is detailed in the *Data Structure Design* section.

# 3.0 Data Structure Design

The data is stored in a noSQL database in Google Firebase using the Cloud Firestore solution offered in Firebase. The database comprises of two collections of documents:

* The *Users* collection
* The *Places* collection

The *Users* collection contains the documents that pertain to each user who has signed up with the application. These documents will be called *User* documents. The table below shows the fields contained in the *User* documents.

|  |  |
| --- | --- |
| **Field Name** | **Field Type** |
| firstName\* | String |
| lastName\* | String |
| dateOfBirth\* | Date |
| address | String |
| country | String |
| email\* | String |
| password\*^ | String |

Fig 3.0.1 Fields of User documents

Fields marked with \* are required fields. Fields marked with ^ are never visible to any user and are encrypted when stored and transmitted via any medium.

The *Places* collection contains the documents that pertain to a specific location on the map. These documents will be called *Place* documents. The table below shows the fields contained in the *Place* documents.

|  |  |
| --- | --- |
| **Field Name** | **Field Type** |
| name\* | String |
| address | String |
| phone | String |
| website | String |
| description\* | String |
| rating | Integer |

Figure 3.0.2 Fields of Place documents

Fields marked with \* are required fields.

# 4.0 Use case realizations

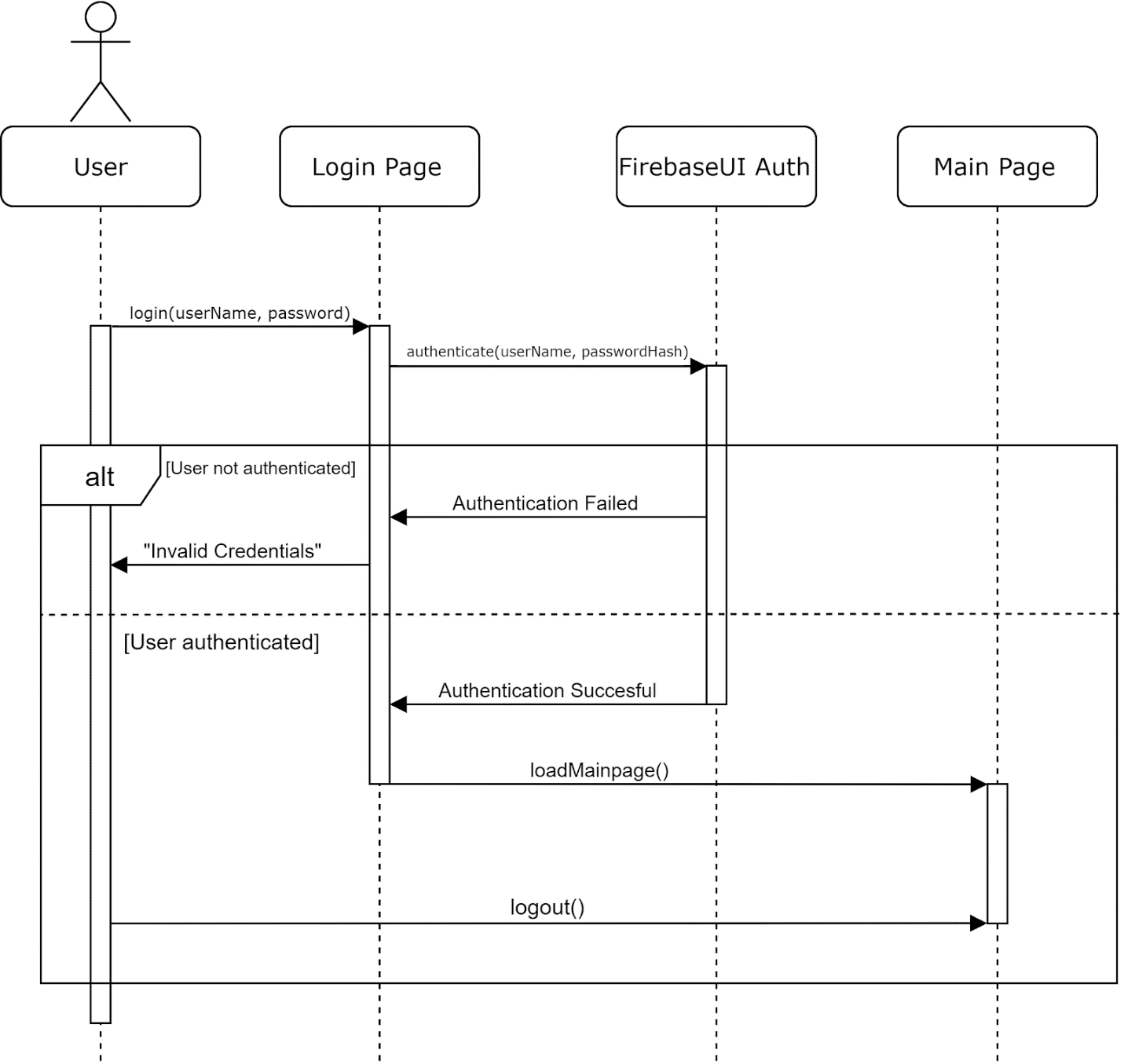
## **4.1 Use Cases**

# 

# 

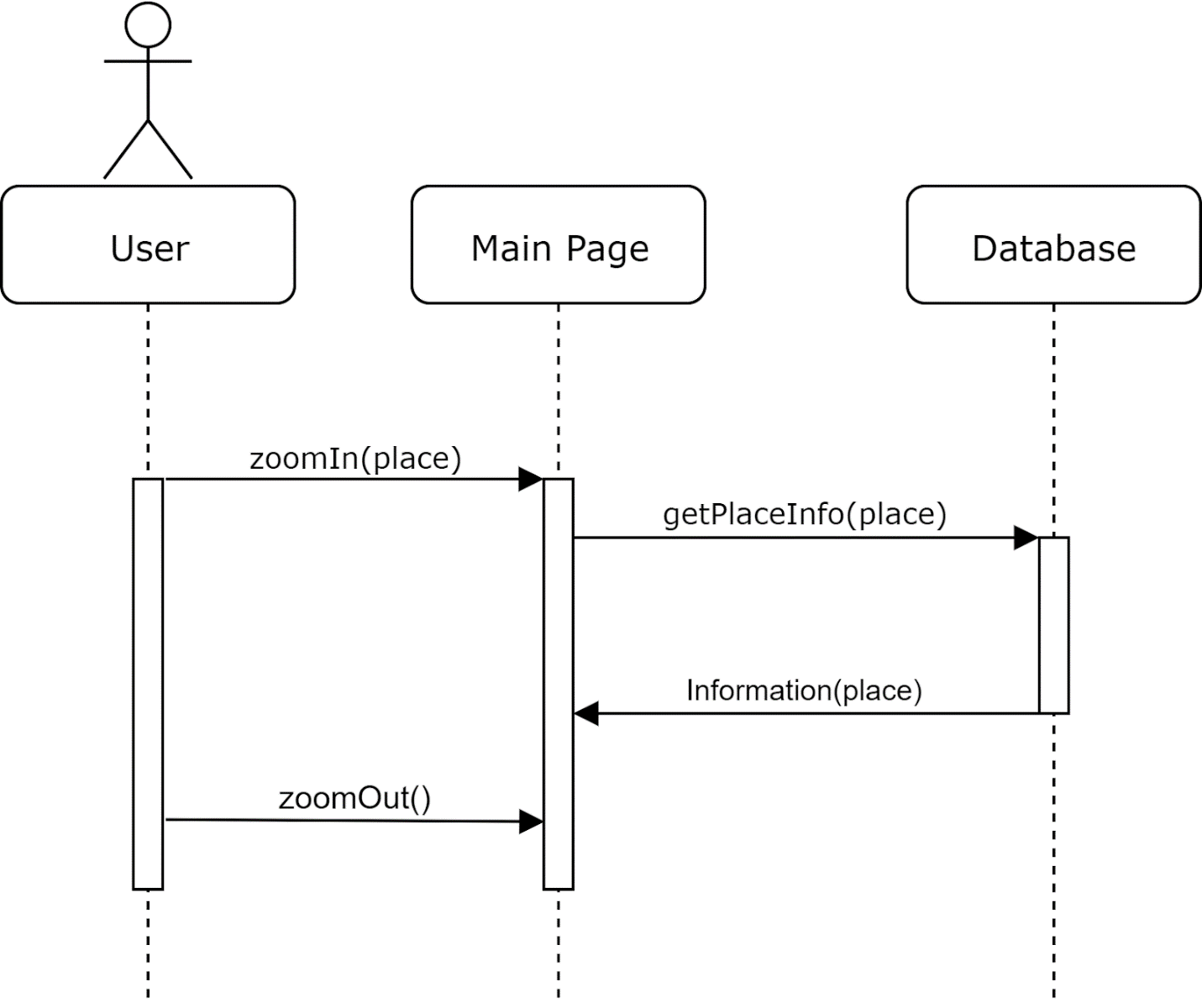
## **4.2 Use Case Realizations**

### **4.2.1 Use Case: User log-in and log-out**



The user enters the user-name/email and password on the login page and then clicks on the log-in button. The credentials are authenticated by the FirebaseUI Auth service provided by Google Firebase. If the authentication is successful, then the user is redirected to the Main page of the application. Otherwise, the user is prompted to re-enter the user-credentials on the Login page. If the user is logged in, the user can log out of the application by clicking on the log-out button on the Main page.

### **4.2.2 Use Case: Zoom in on a place and zoom out**



Once the user is on the Main page, the user can zoom in on a place on the map in the Main page. The user can do this by scrolling the mouse wheel. When the user zooms in on a place, information about the place zoomed in is fetched from the Database. This information is displayed to the user on the Main page. When the user zooms out on the map using the mouse wheel, the Main page clears the information displayed to the user. The Main page would display information relating to the area the map is currently displaying.

# 5.0 Interface Design

Below are the anticipated interface designs

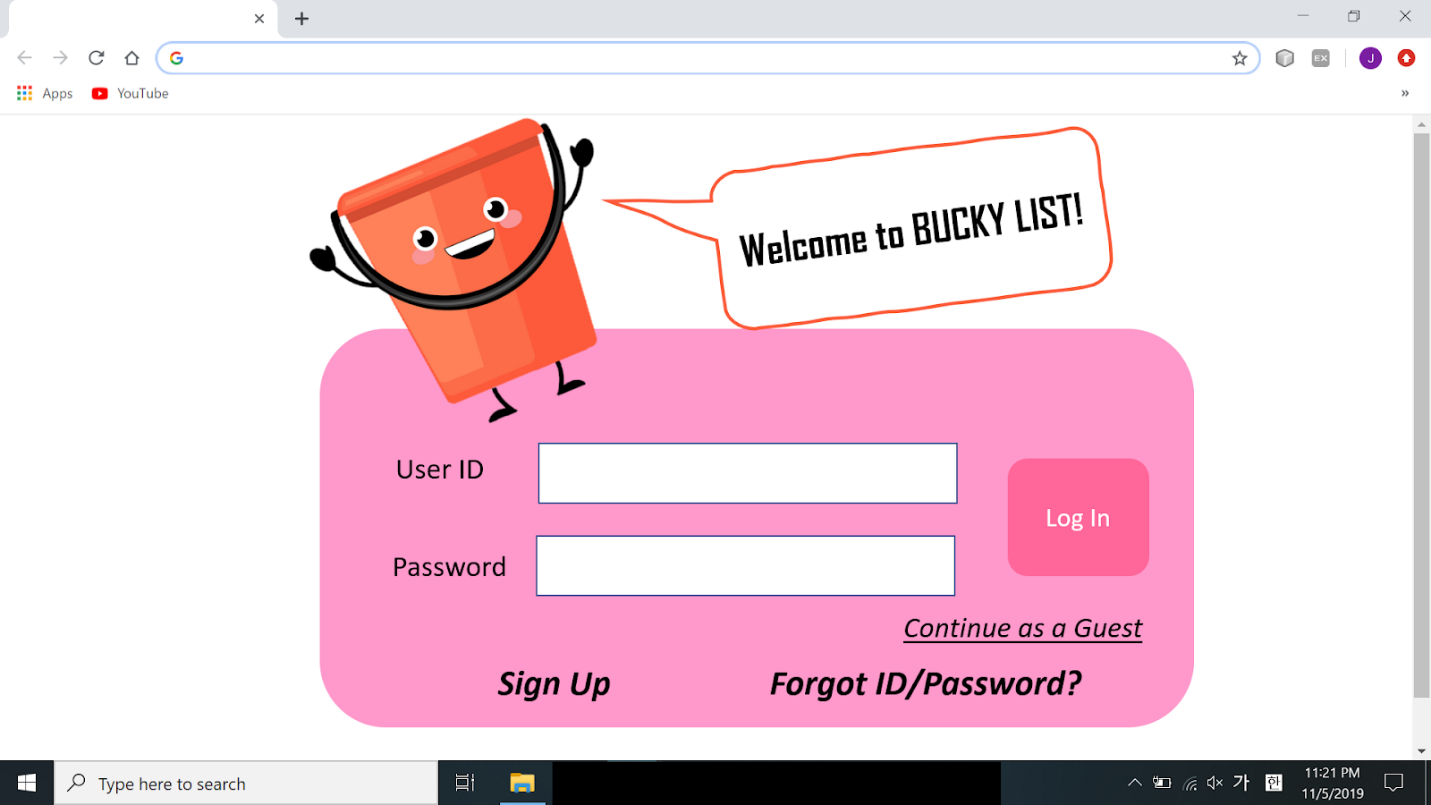
.

Figure 1 Main page

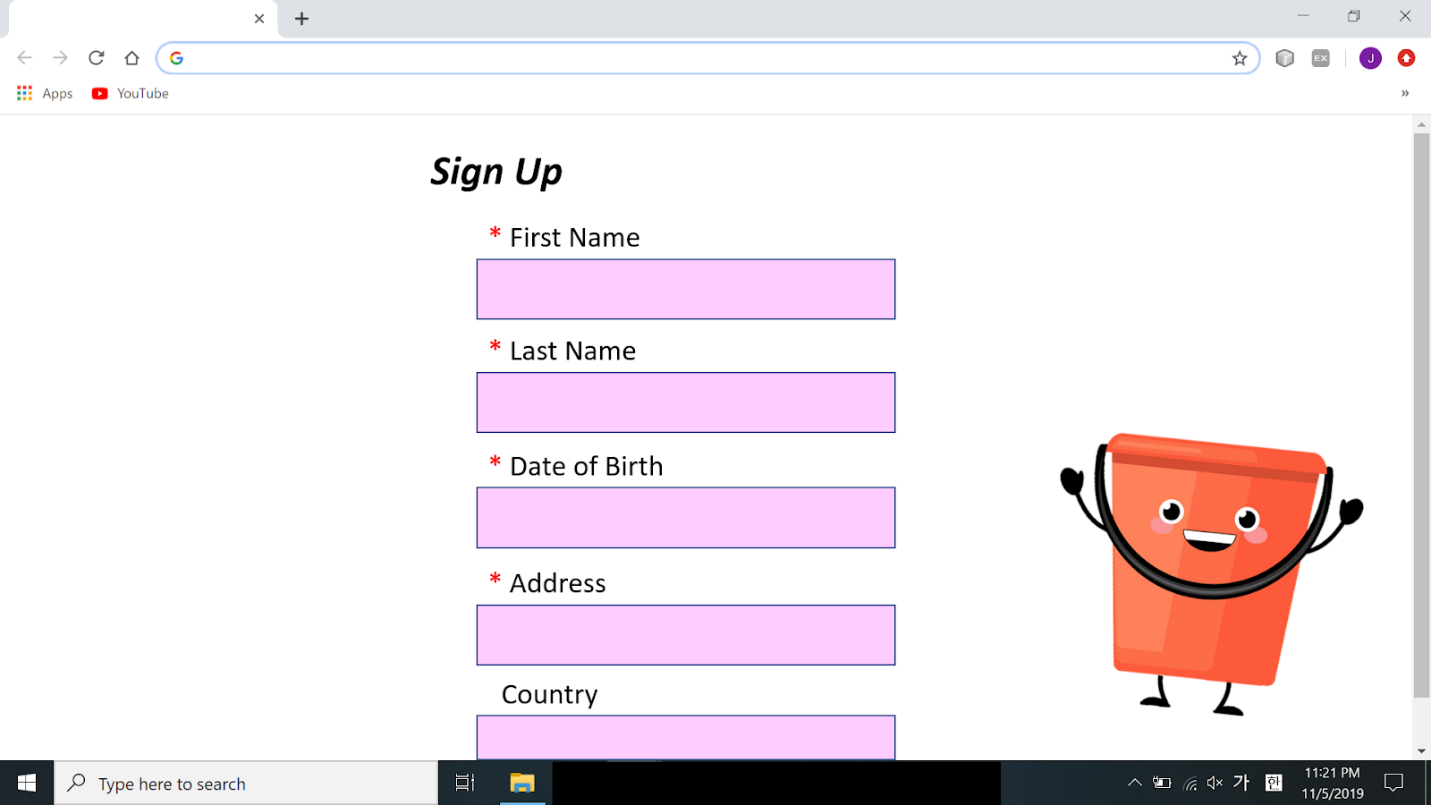


Figure 2 Sign up page

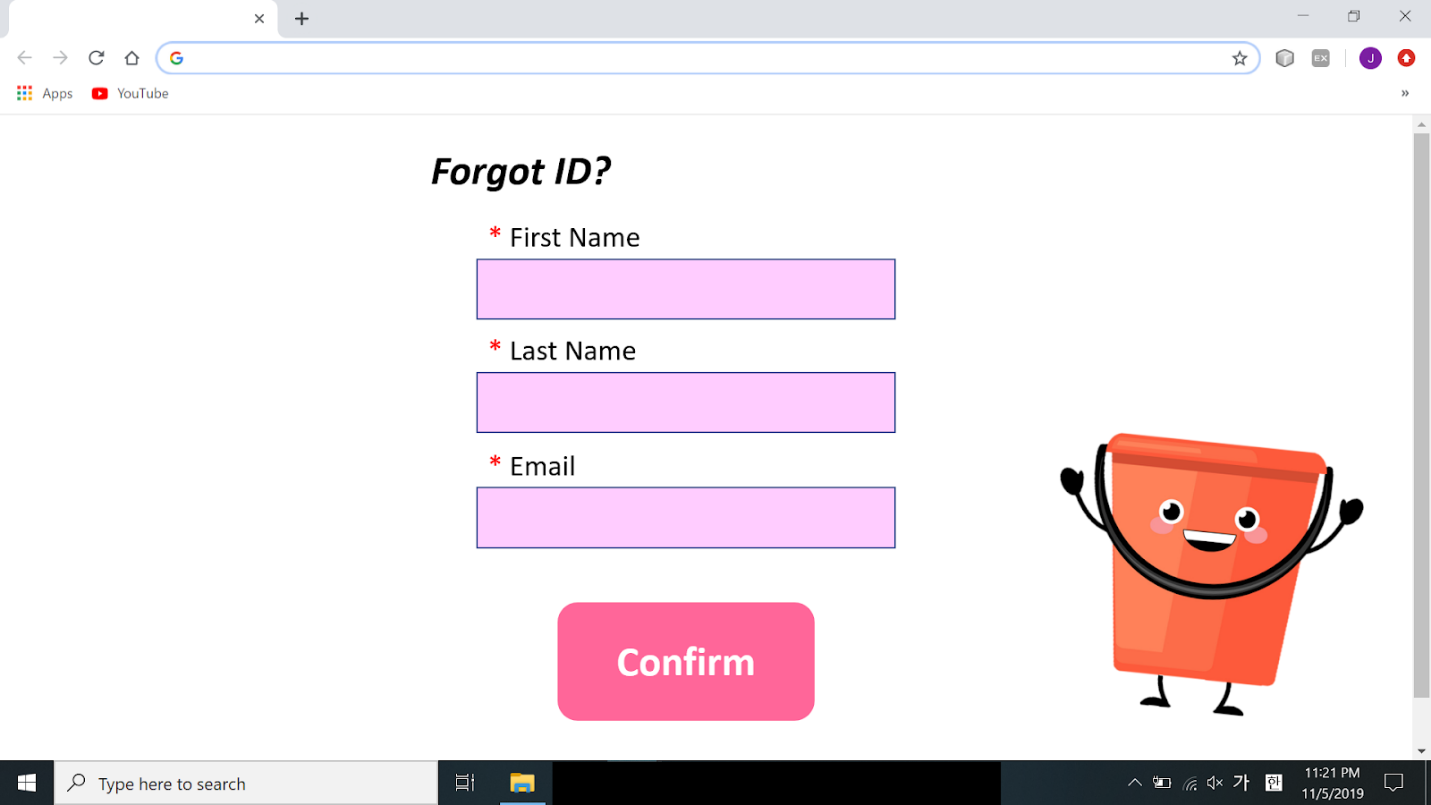


Figure 3 Find ID / Password page



Figure 4 World map page

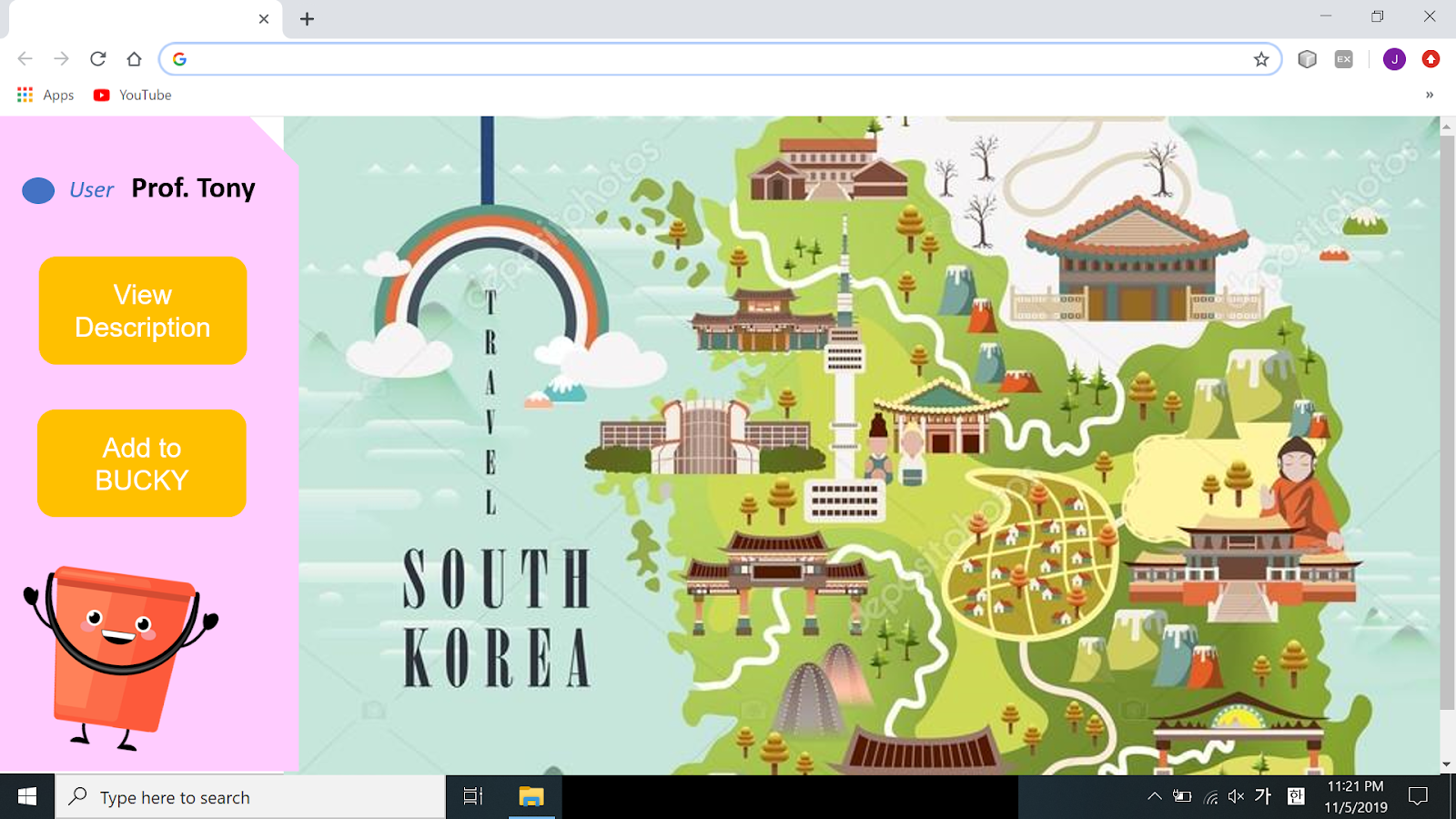


Figure 5 Korea page

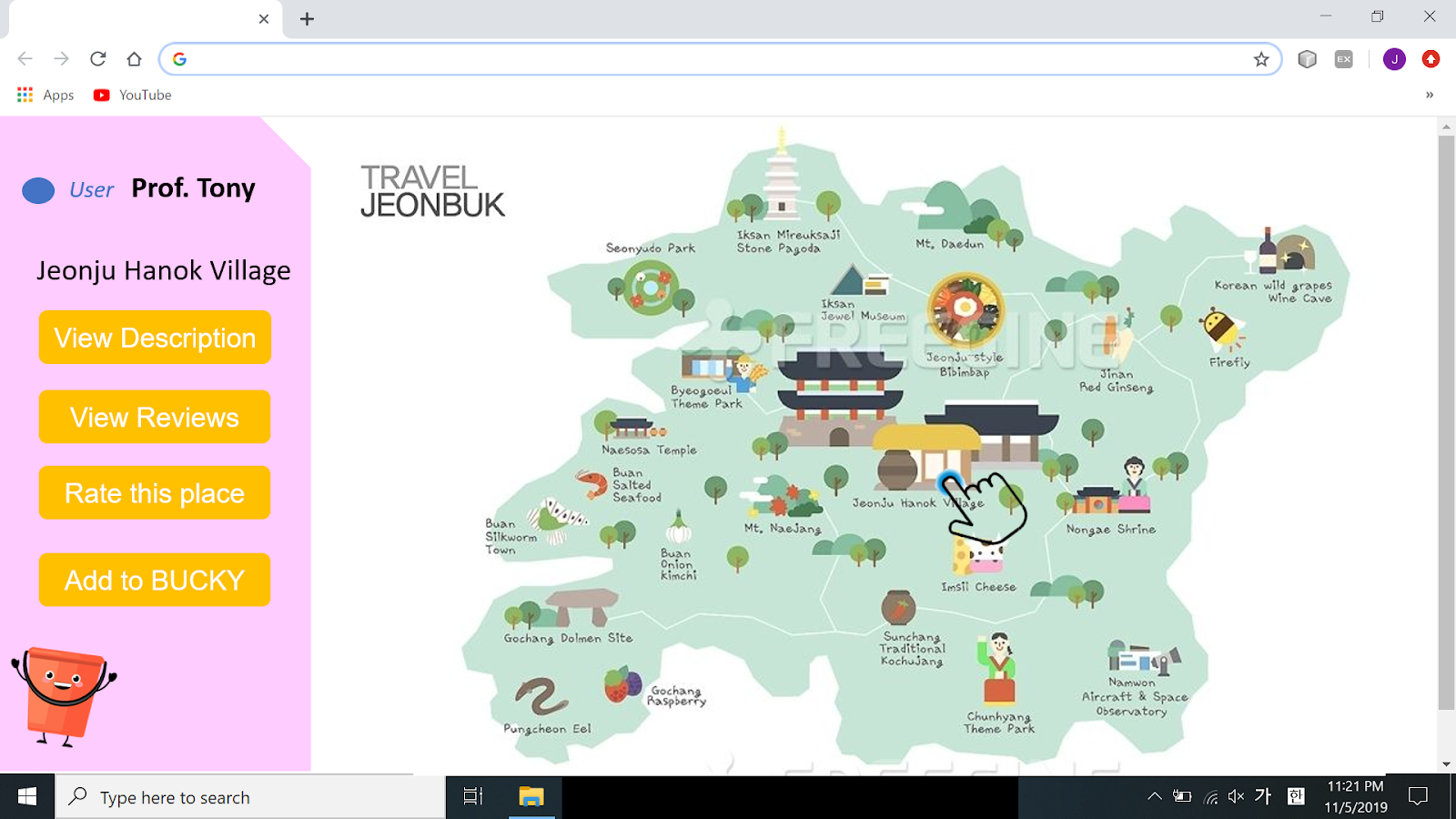


Figure 6 View specific place page

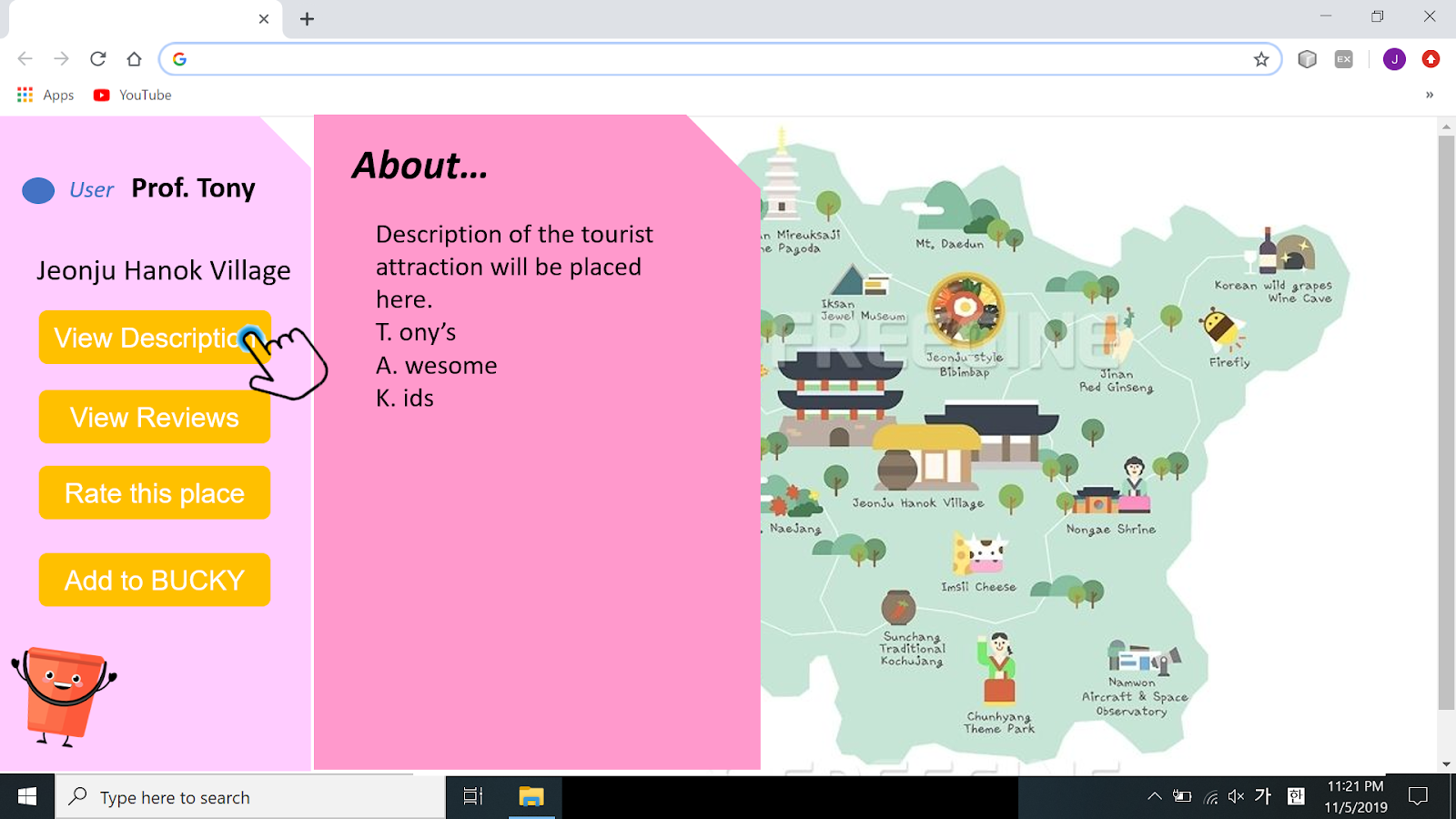


Figure 7 View description page

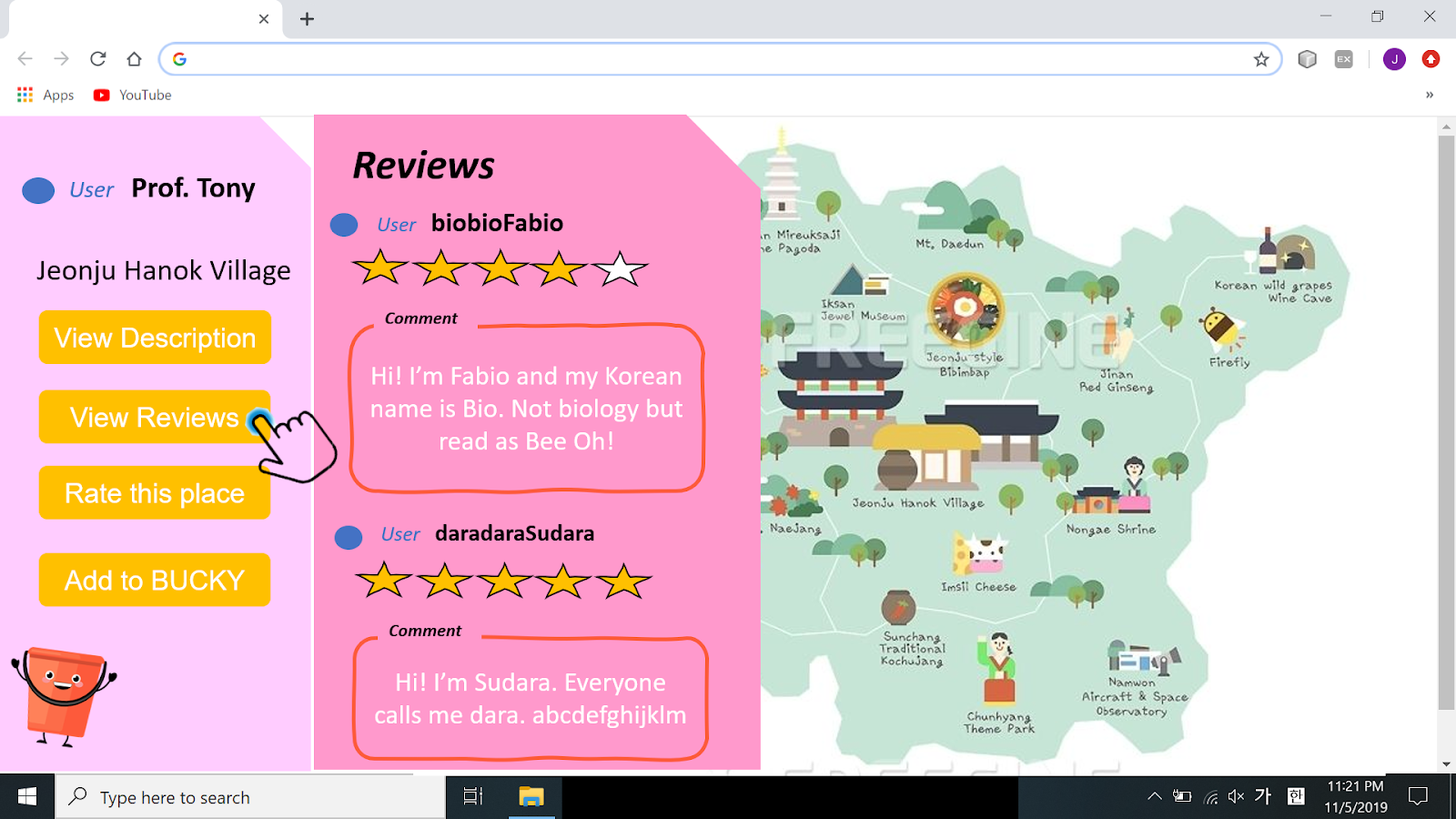
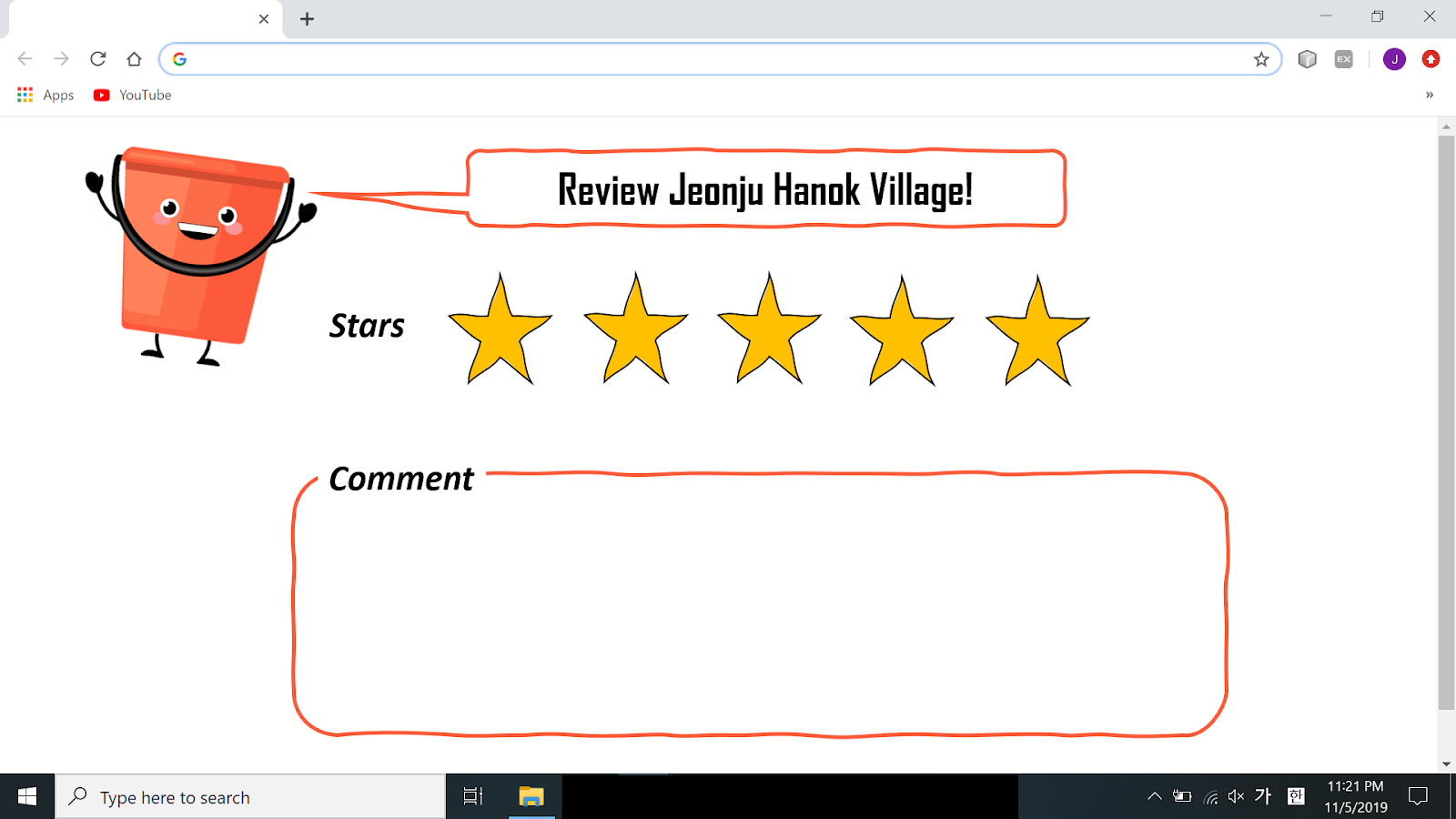


Figure 8 View reviews page

Figure 9 Rate the place page

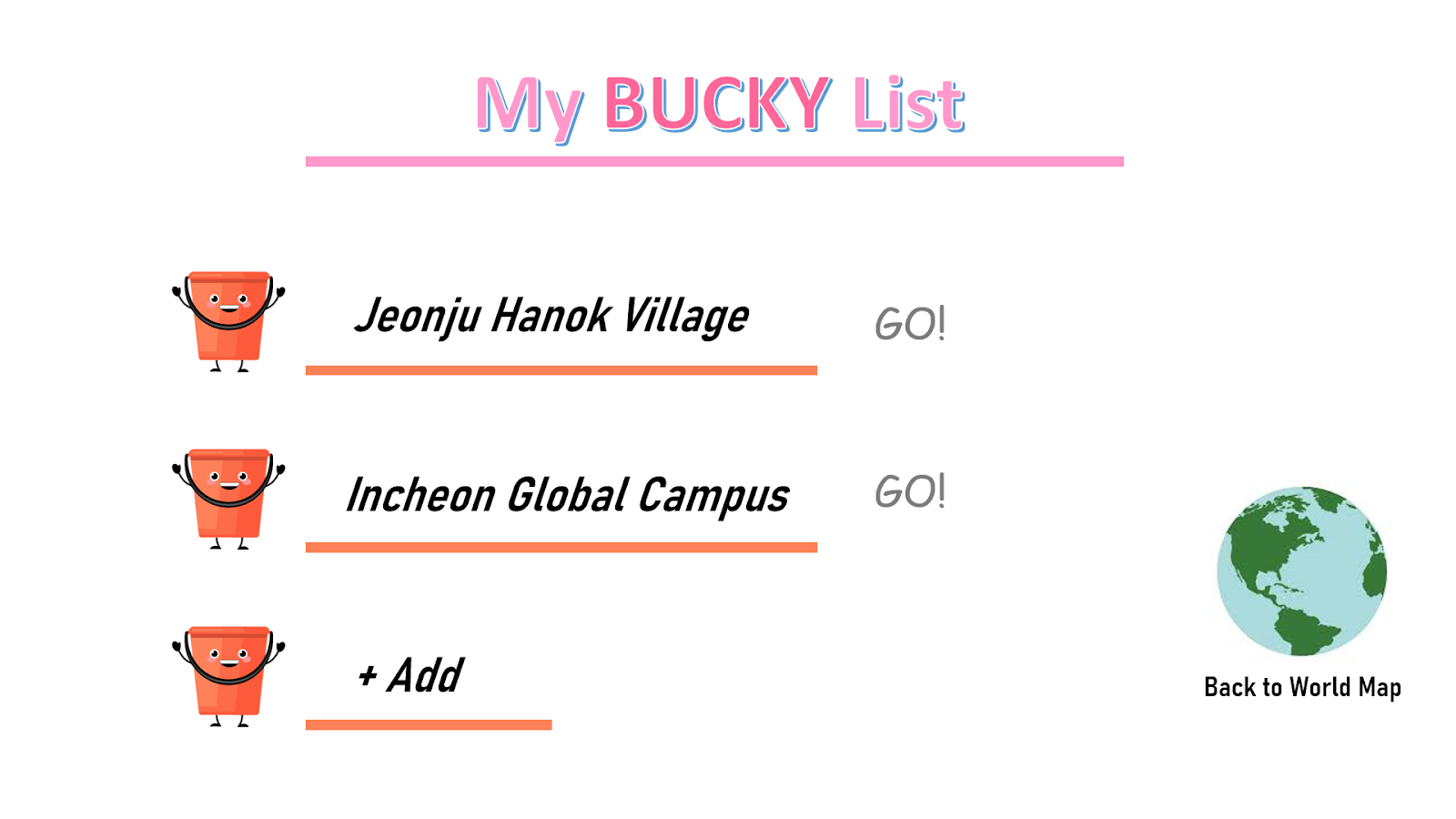


Figure 10 BUCKY List page

**6.0 Help System Design**

There is no help system to be implemented in the initial sprints of the project. A later version of this document will contain a detailed Help System Design section.