Contents

[**Interview Assignment Answers (Ian Pee)** 2](#_Toc41211123)

[1.0 Development and deployment of a login service 2](#_Toc41211124)

[1.1 Actors 2](#_Toc41211125)

[1.2 Description 2](#_Toc41211126)

[1.3 Use Case Diagram 2](#_Toc41211127)

[1.4 Sequence Diagrams 3](#_Toc41211128)

[1.4 Screens 7](#_Toc41211129)

[1.5 Client-side Validation 9](#_Toc41211130)

[1.6 Firebase - Realtime Database & Authentication 9](#_Toc41211131)

[1.7 Test Data 9](#_Toc41211132)

[1.8 Test Cases & Results 10](#_Toc41211133)

[2.0 Design and planning for an application involving a recent technology 13](#_Toc41211134)

# Interview Assignment Answers (Ian Pee)

## Development and deployment of a login service

### 1.1 Actors

User – Involved in the login service. The user can sign in, sign out, register new account and edit their personal details while signed in.

### 1.2 Description

This login service is developed using HTML, Bootstrap Framework, jQuery and Firebase Realtime Database as the server. I have created **5** simple modules each with its own MVC. These modules made up the whole login service. The modules are *login, register, home, forgotPassword and resetPassword.*

### 1.3 Use Case Diagram

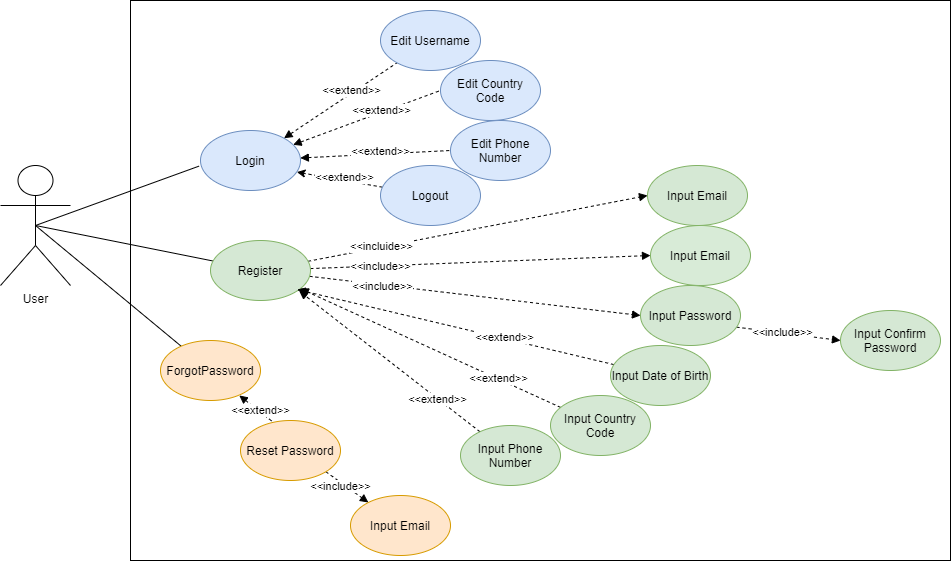


Figure 1: Use case diagram for the login service.

### 1.4 Sequence Diagrams

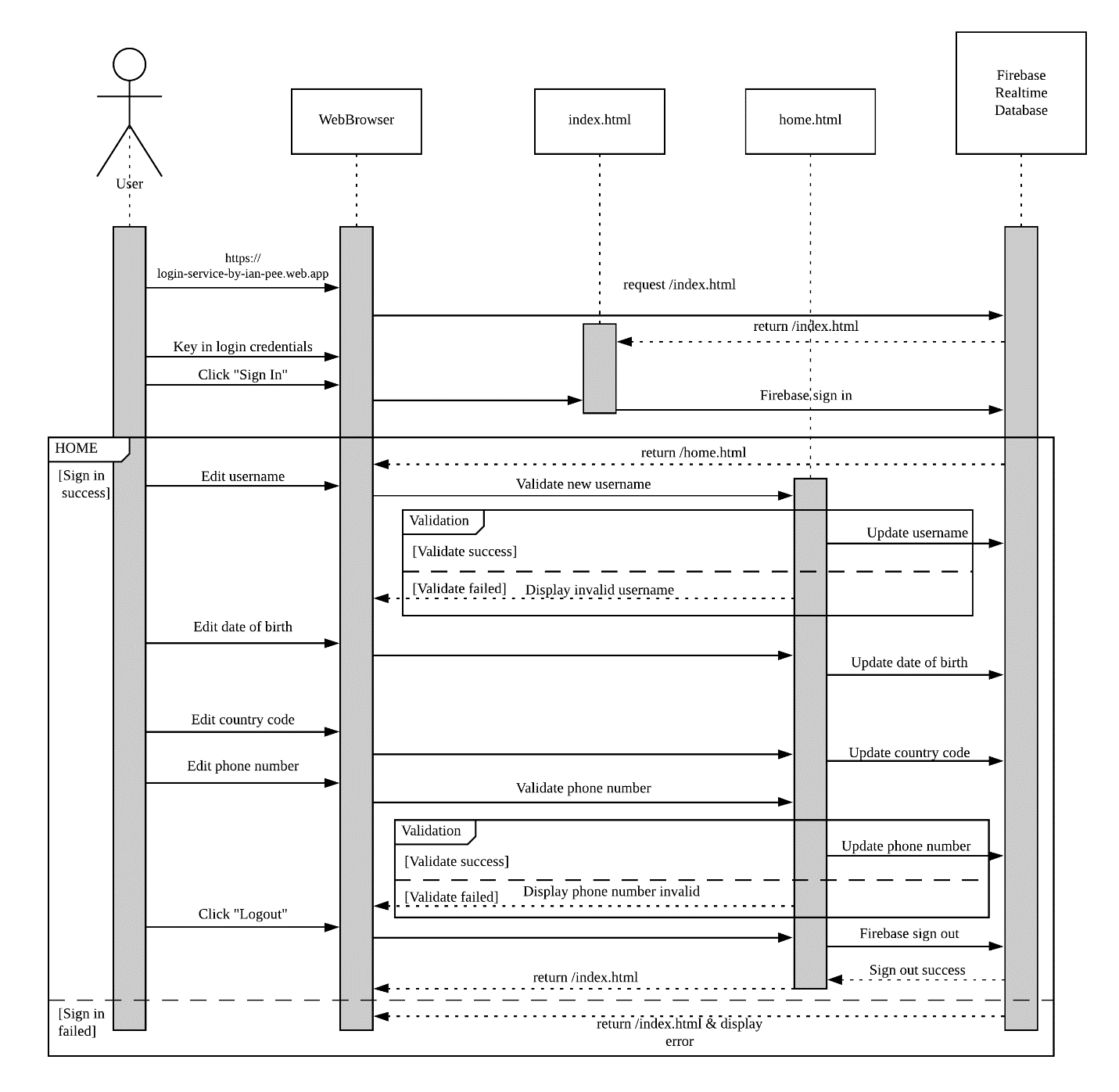


Figure 2: Sequence diagram for login module & home module.

Figure above depicts the sequence diagram for *login module & home module*. There are 2 sections in the diagram, the section outside of the “HOME” box represents the login module. When the user is successfully authenticated, the user can use the home module which is in the “HOME” box.

The user can sign in using their registered email & password or sign in with their Google account. If they sign in with their Google account for the first time, the app will automatically register the Google account in the Firebase and initialize the username with the name from their Google account.

In the *home module,* the user can edit their personal details.

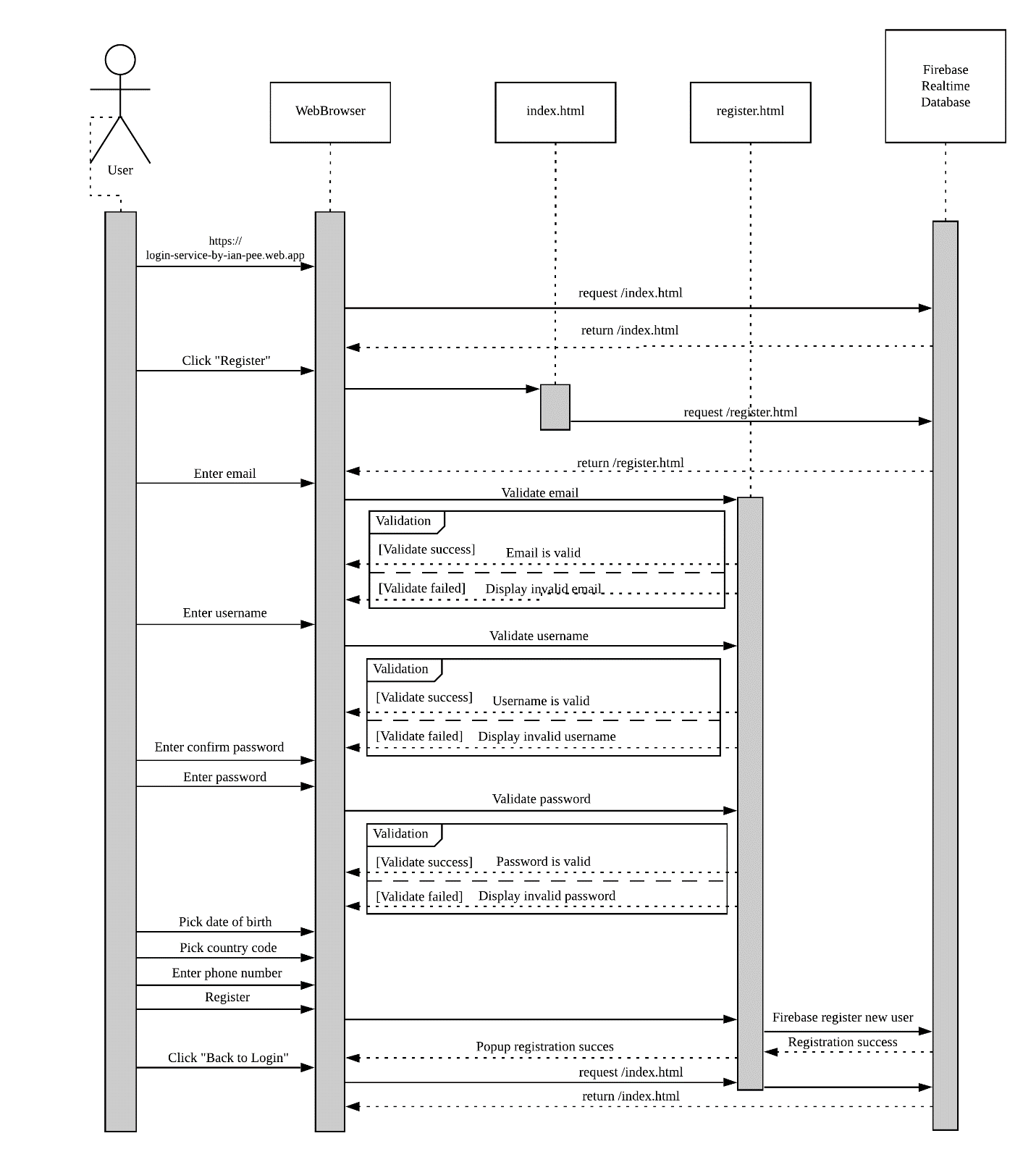


Figure 3: Sequence diagram for register module.

Figure above depicts the sequence diagram for the *register* module. The *register* module can be accessed from the *login* module by clicking “Register”. Here, the new user can register their account with an email address, username, password and optional info such as date of birth, country code and phone number. Once submitted and is registered in Firebase Realtime Database, a popup will show and allow to user to return to the login page.

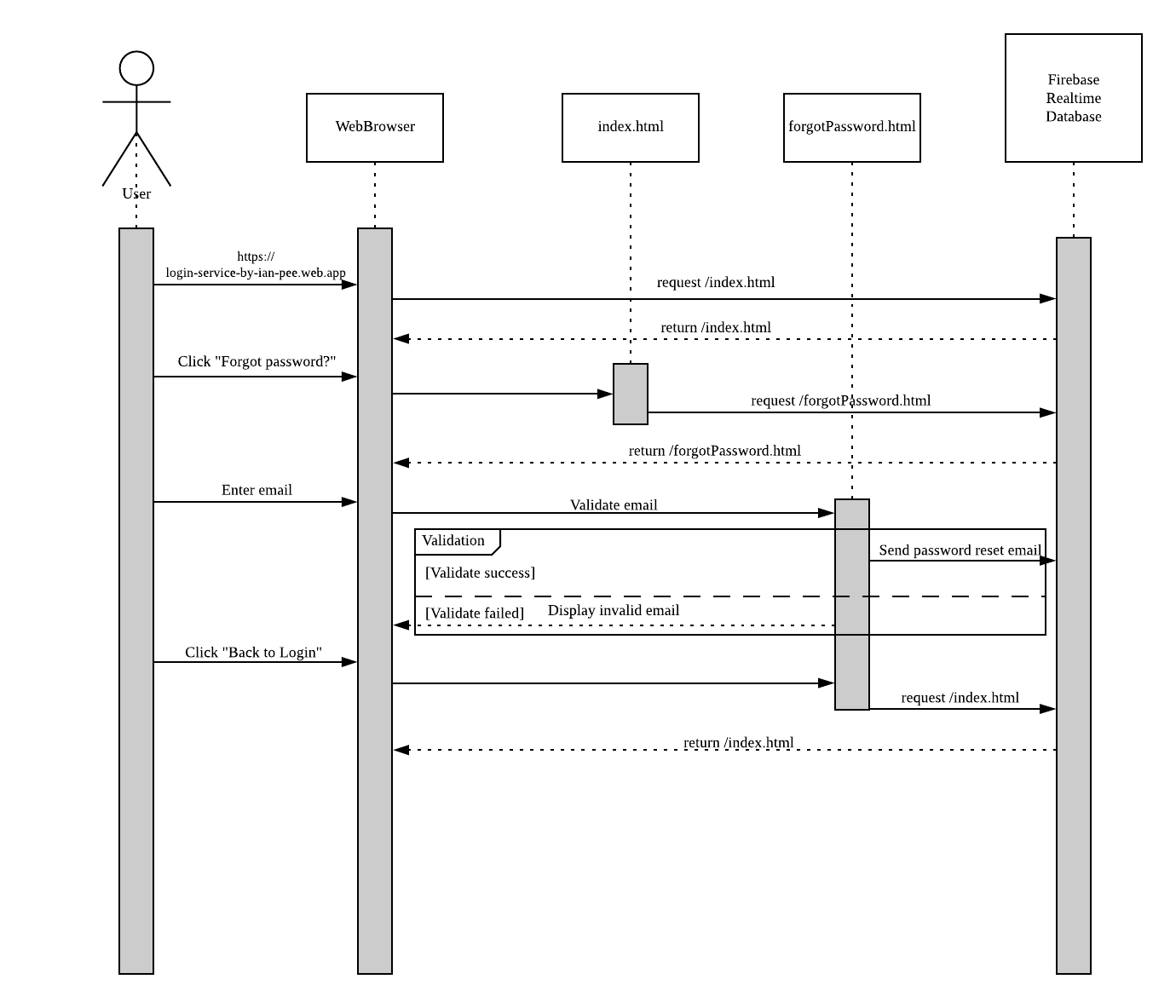


Figure 4: Sequence diagram for forgotPassword module.

Figure above depicts the sequence diagram for the forgotPassword module. The forgotPassword module can be accessed from the login module by clicking on “Forgot Password?” Here, the user can request for a password change link to their email. The link that is sent to their email will lead to the next module, resetPassword module.

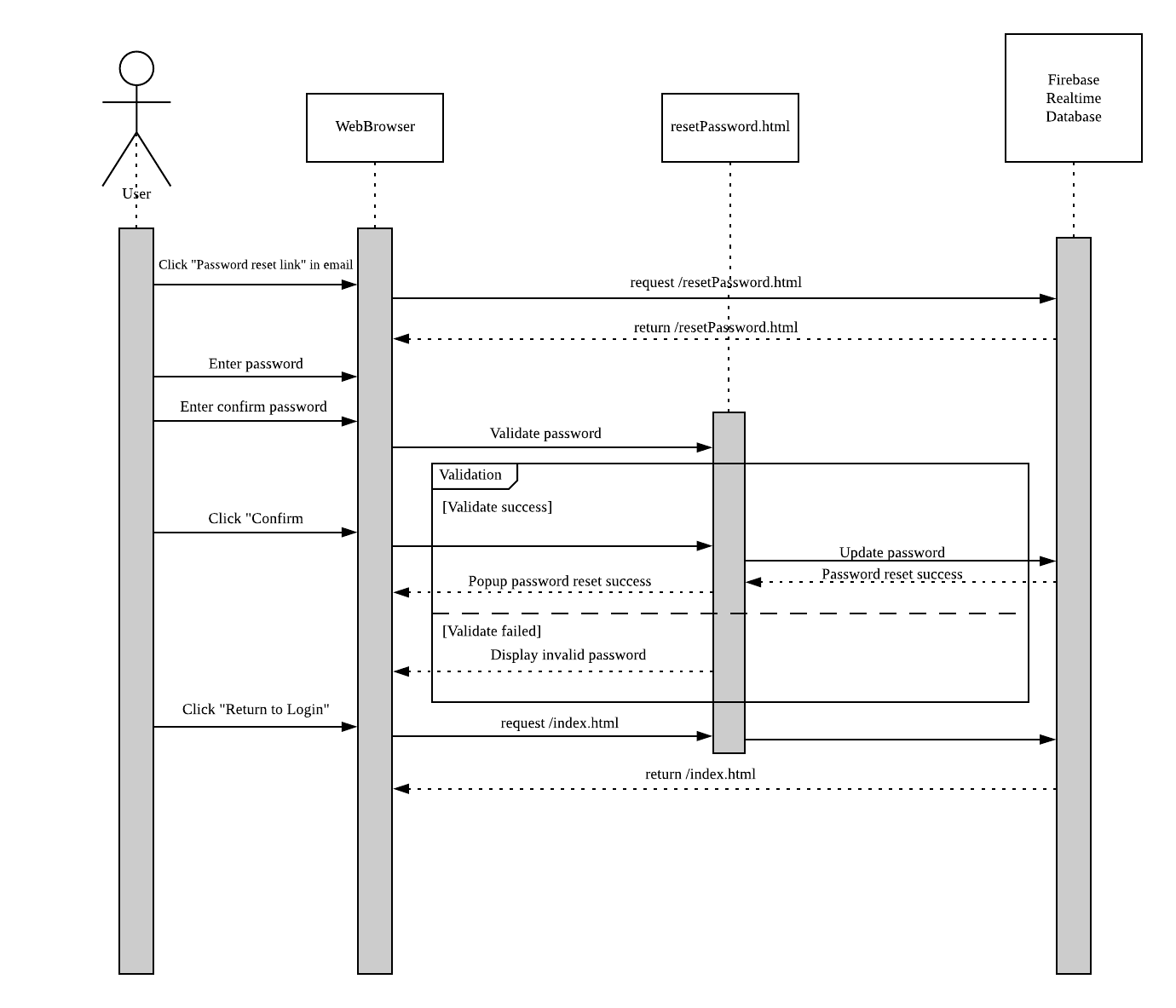


Figure 5: Sequence diagram for resetPassword module.

Figure above depicts the sequence diagram for the *resetPassword* module. The only way to access this module is via the link sent to the user’s email box from *forgotPassword* module. Here, the user can assign a new password to their account. Besides, this module requires an *oobCode* from the password reset link that is generated from Firebase Authentication API to work.

### 1.4 Screens

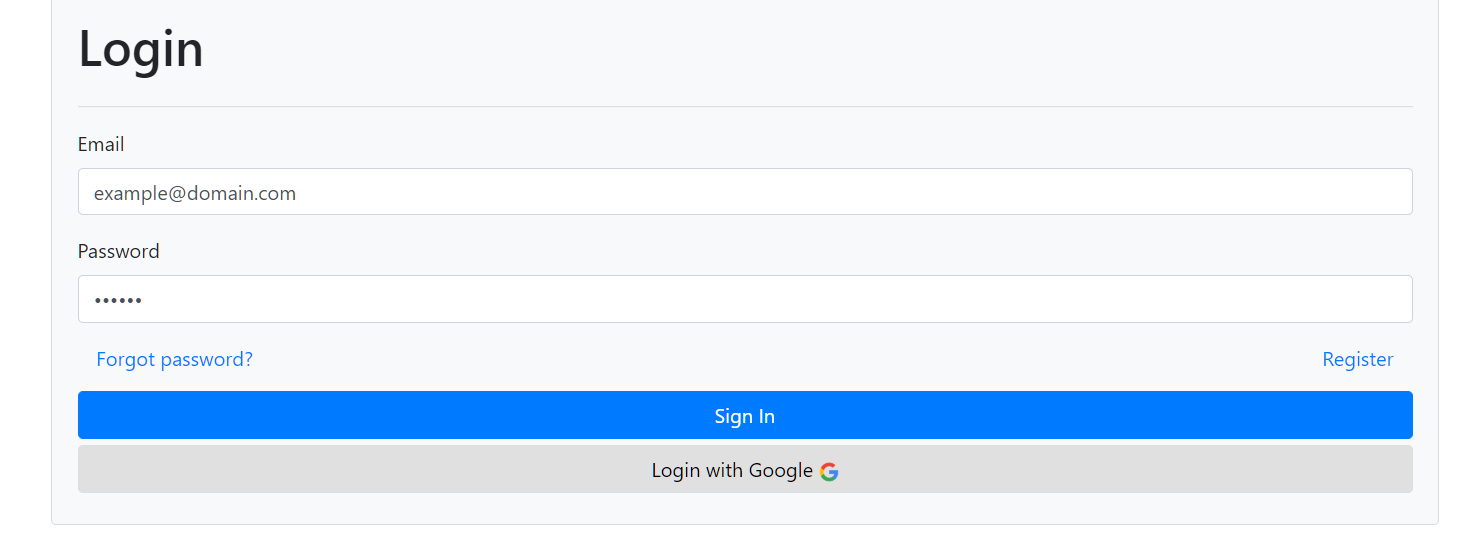


Figure 6: Login page.

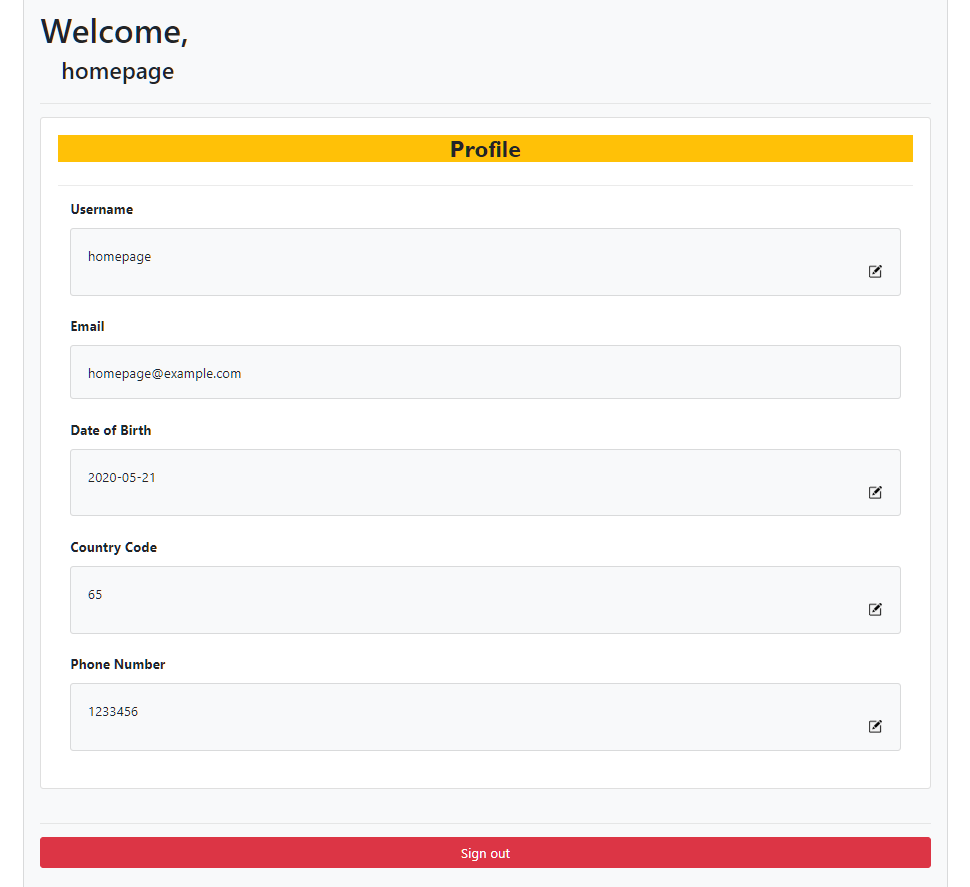


Figure 7: Home page.

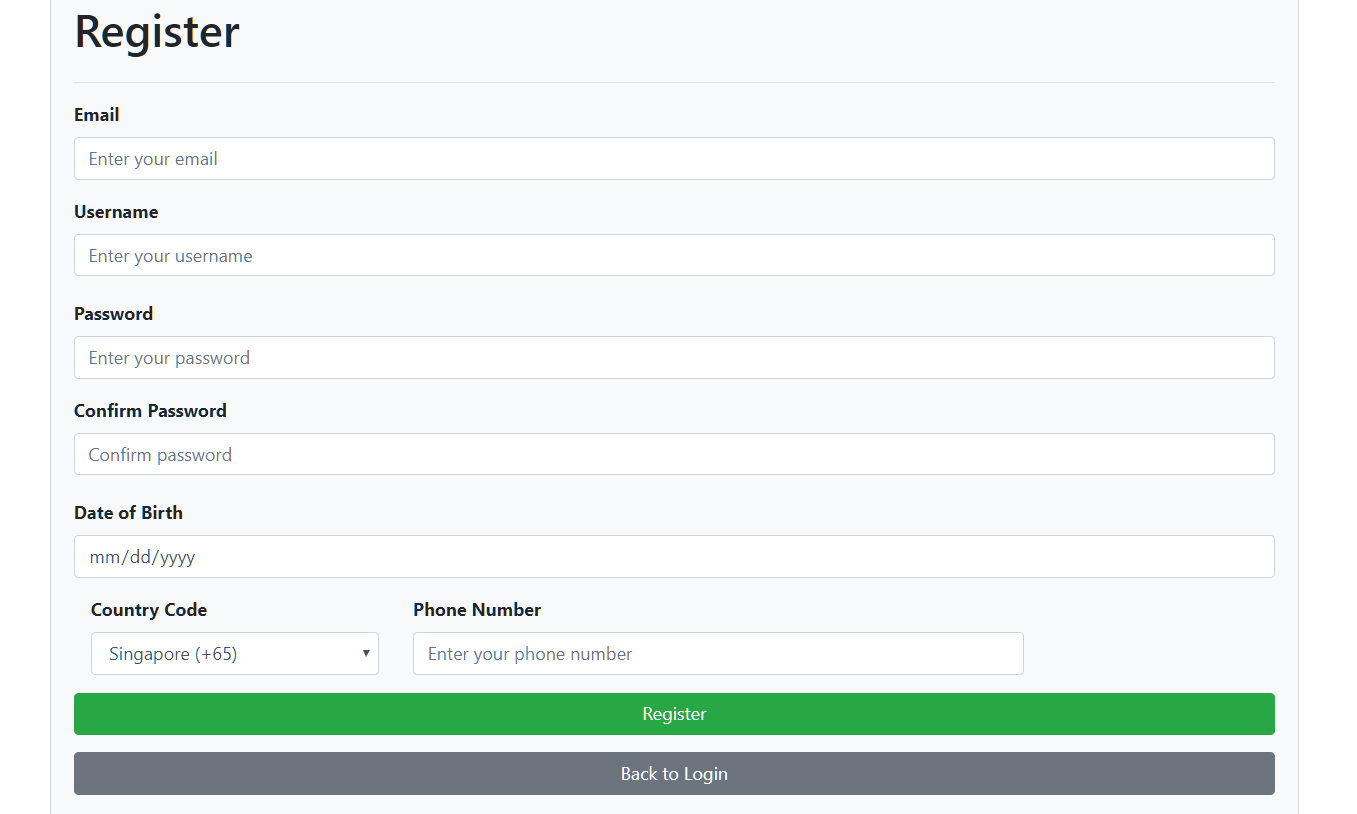


Figure 8: Register page.

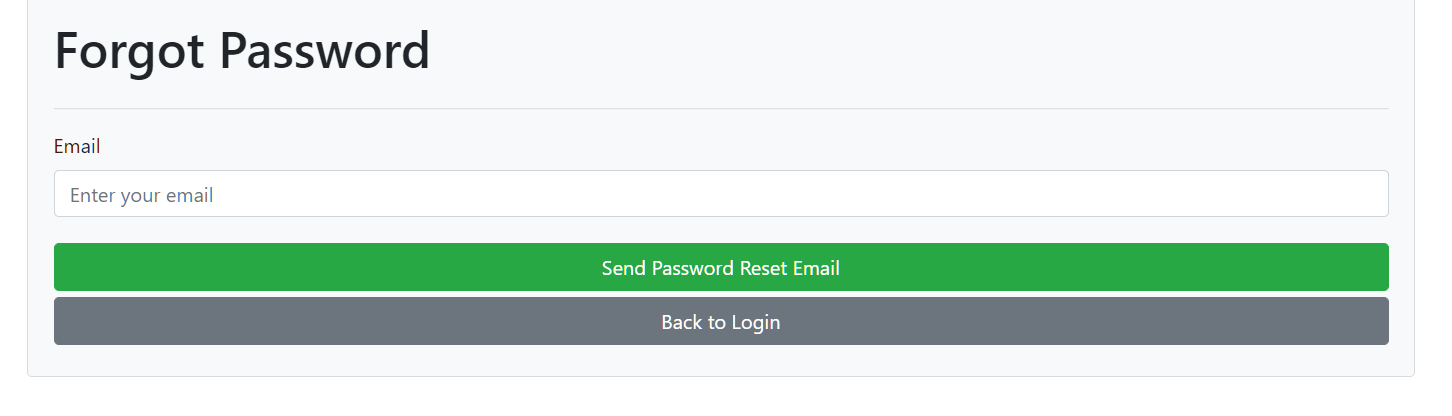


Figure 9: Forgot password page.

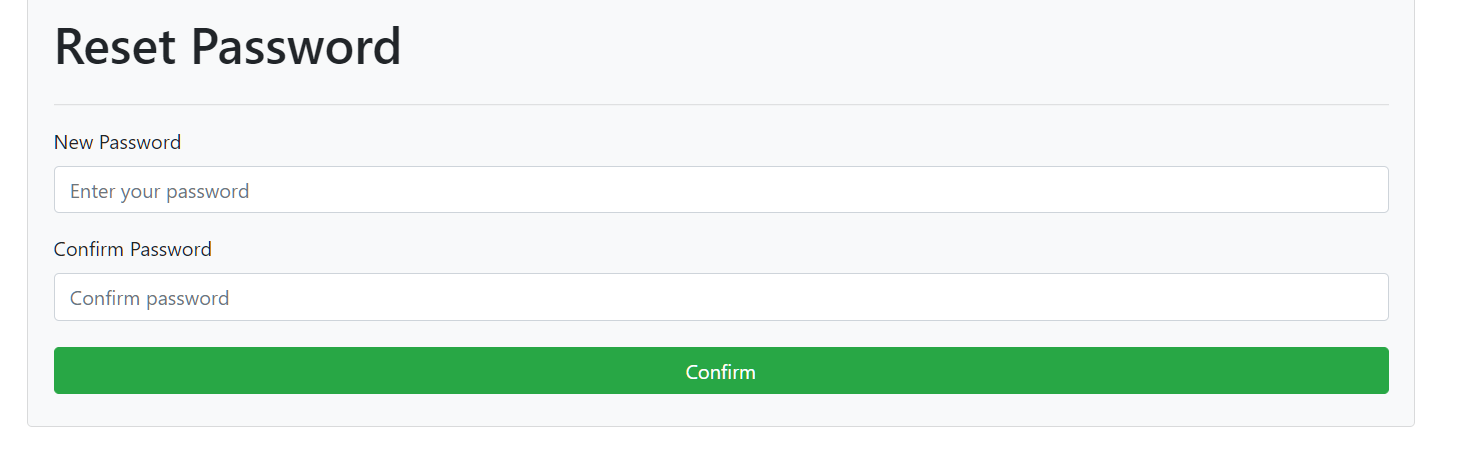


Figure 10: Reset password page.

### 1.5 Client-side Validation

Every time a user registers a username, password, email or a phone number, it will get validated before being able to set or update in Firebase. The validation script *clientSideValidation.js* provides functions for other modules to utilize such as, *validatePassword(), validateUsername() and validateCountryCode().*

These functions normally perform a combination of 1 to 3 things,

*1)* to check if the input is in accordance to a regex, and,

*2)* to check if the data is valid for database storage, or

*3)* to check with the Firebase Realtime database on the existence of the data (e.g. username taken).

### 1.6 Firebase - Realtime Database & Authentication

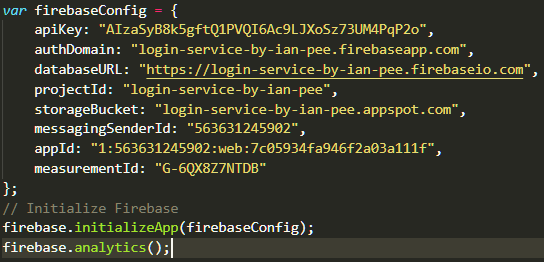


Figure : firebaseInit.js

Every time a module is loaded, it will first initialize the Firebase main API using the script above. Any Firebase API such as the ones used in this project, Firebase Database & Authentication, must be included before this script for Firebase to run successfully. T

The risk of having this script exposed on the client side is dangerous, as attackers can flood the database with data and possibly crash it. The correct way of protecting our data should be done on the server side, which is called server-side validation. Firebase has its own server rules that we can set to reject any incoming data and protect the integrity of the database.

### 1.7 Test Data

|  |  |
| --- | --- |
| Account Details | Data (valid/invalid) |
| Email | [homepage@example.comm](mailto:homepage@example.comm) / [doesNotExist@example.com](mailto:doesNotExist@example.com) |
| Password | Homepage1 / homepage1 |
| Username | homepage |
| Date of Birth | 2020-05-21 |
| Country Code | 65 |
| Phone Number | 1233456 |

Table 1: Test Data #1 #TD1

|  |  |
| --- | --- |
| Registration Details | Input (valid/invalid/invalid…) |
| Email | [me@example.com](mailto:me@example.com) / me\_example.com |
| Username | Me\_example / me example |
| Password | MeExample1 / meExample1 / meexample1 |
| Confirm Password | MeExample1 / meExample1 / meexample1 |

Table 2: Test Data #2 #TD2

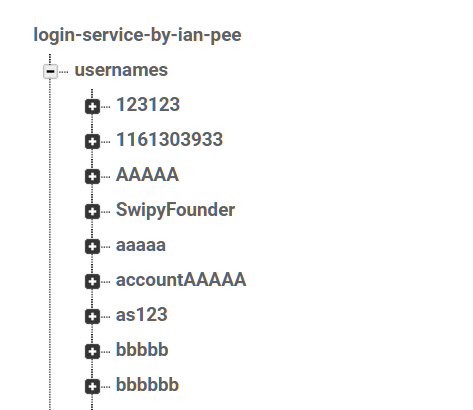


Figure : Current database snapshot for registered usernames. (Notice username “aaaaa”, used in Test Case #**TC05**)

### 1.8 Test Cases & Results

|  |  |  |
| --- | --- | --- |
| Test Cases | Expected Output | Actual Output |
| #**TC01,TD1**  Login using #TD1 with valid password. | Login success, redirect to home page, with corresponding data values as indicated in #TD1. | *As expected,* |
| #**TC02,TD1**  Login using #TD1 with invalid password. | Login failed, error displaying “Wrong password”. | *As expected,* |
| #**TC03,TD1**  Login using #TD1 with invalid email. | Login failed, error displaying “Couldn’t find your account” | *As expected,* |
| #**TC04,TD1**  From #T01, edit the username from “homepage” to “newHomepage” | Usernames change to “newHomepage” | *As expected,* |
| #**TC05,TD1**  Edit username again, this time with an existing username in database “aaaaa”. | Cannot update username and shows username already taken. | *As expected,* |
| #**TC06,TD2**  Register with valid details. | Registration popup successful. | *As expected,* |
| #**TC07,TD2**  Register with existing email. | Displays error “email already registered” | *As expected,* |
| #**TC08,TD2**  Register with invalid  username | Displays error “username requirements” | *As expected,* |
| #**TC09,TD2**  Register with invalid password | Display error “password requirements” | *As expected,* |
| #**TC10,TD**  Send password reset email to registered email. | Popup successful sent email. | *As expected,* |

## 2.0 Design and planning for an application involving a recent technology