# JS Apps Exam – Destination App

You are assigned to implement a **Web application** (SPA) using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend (Firebase). The **app** that keeps **users** (destinations) and manages **them**. Users can **register**, **login**, **logout**, view a page with **all treir destinations**, **create** a new one, **edit** and **delete** their own destinations, view a **detailed** page of a destination.

You are **allowed** to use libraries like **jQuery**, **Handlebars** and **Sammy**. Frameworks and libraries like React, Angluar, Vue are **not permitted**.

## Create a REST Service

Use any cloud-based databaseand create application to keep your data in the cloud.

Create a collection called **destinations.** Each **destination** has a **destination** **name**, **city**, **imageUrl**, **duration, departure date**.

## HTML and CSS

You have been given the web design of the application as **HTML** + **CSS** files.

* Initially all views and forms are shown by the HTML. Your application may **hide**/**show elements** by CSS (display: none) or **delete**/**reattach** from and to the DOM all unneeded elements, or just display the views it needs to display.
* You may render the views/forms/components with **JavaScript** or **Handlebars**.
* You are **allowed** to add **attributes** to any HTML elements.

**Important**: Don’t change the elements’ **class names** and **ids**. Don’t rename form fields /ids. You may modify href **attributes** of links and add action**/**method **attributes** to **forms**, to allow the use of a routing library.

1. **Client-Side Web Application**

**Design** and **implement** a client-side front-end app (SPA). Implement the functionality described below.

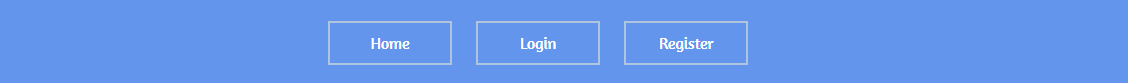
### Navigation Bar (10 pts)

Implement a **navigation bar** for the app: navigation links should correctly change the current screen (view).

* Clicking on the links in the **menu** or **individual** links should display the view behind the link (views are sections in the HTML code).
* **The Logged-in** user navbar should contain the following elements: the **[Home]** - link to the home page, [**Destinations**] a **link** to all Destinations page, [Add +] link to Create Destination page, the user caption ("Welcome, {email}") link to user Profile page and [Logout] link.

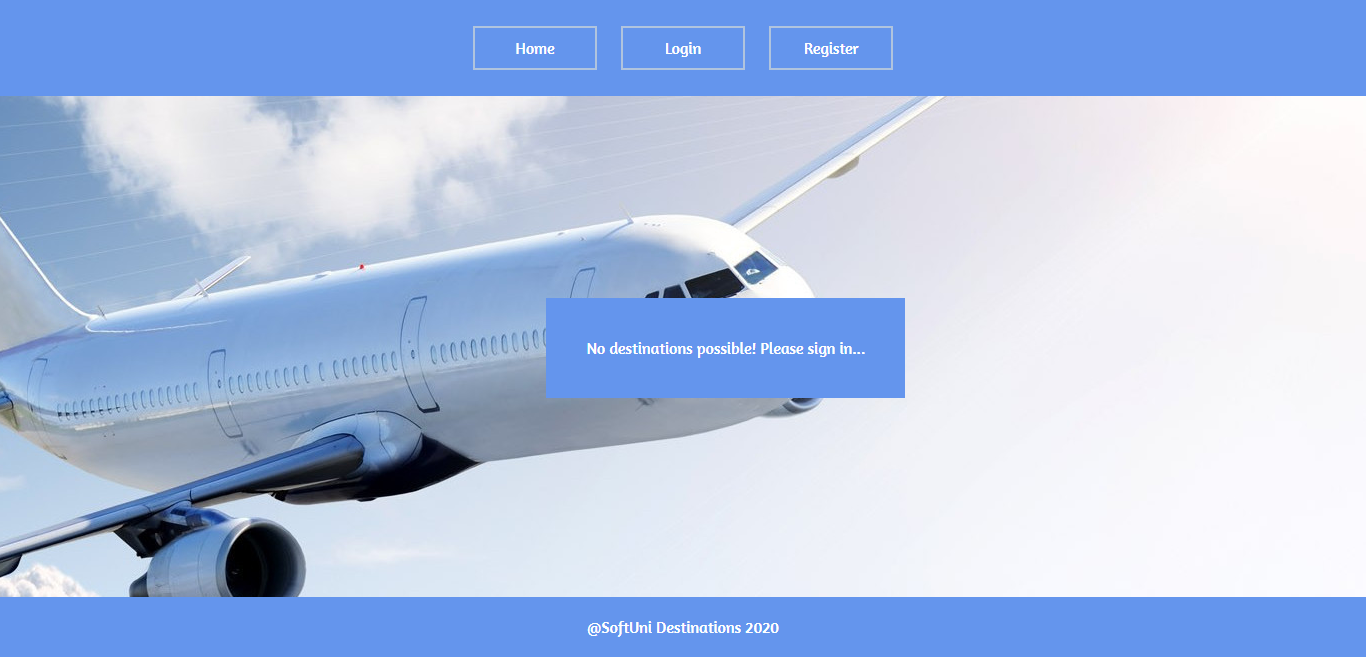
****

* The **guest** users navigation bar should contain the following elements: **Home, Login, Register** (hide the whole div with class "right-container")



**Home Page (Guest) (5 Pts)**

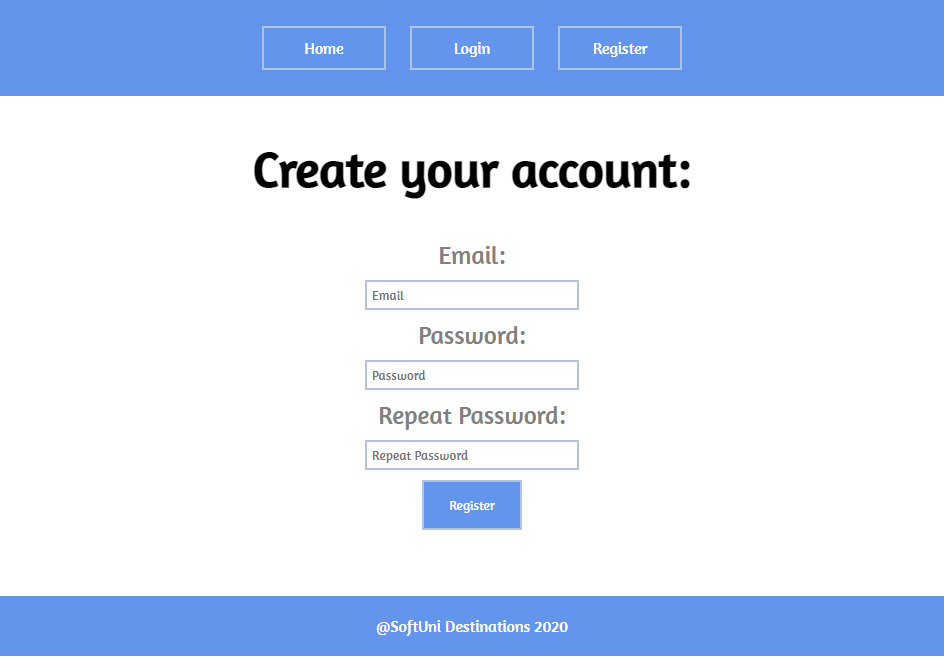
The initial page (view) should display the **guest** **navigation bar** + **Guest Home Page** + Footer.



### Register User Screen (5 pts)

By given **email**, **password and repeat password** the app should register a new user in the system.

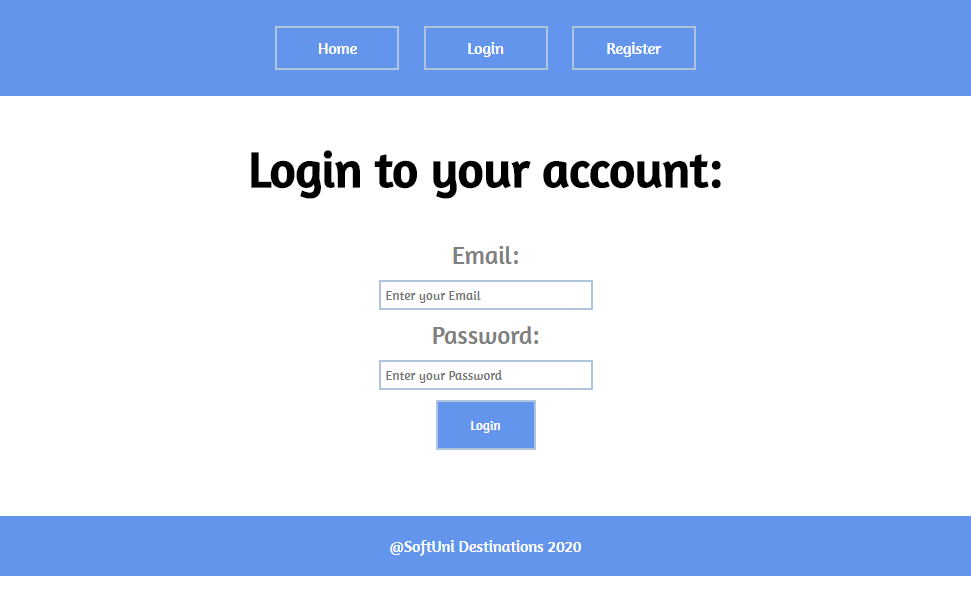
* After a **successful registration**:
* a notification message “User registration successful.” should be displayed (Bonus\*)
* the user should be **redirected** to the home view (already logged in).
* You **need** to validate the **input**. An email **should** be a **valid email string**. Passwords **input** fields shouldn’t be **empty**. Both passwords **should** match.
* In case of **error** (eg. invalid email/password):
* an appropriate error **message** should be displayed (Bonus\*)
* the user should be able to **try** to register again.
* Keep the user session data in the browser’s **session storage/local storage**.
* Clear **all** input fields after **successful** register.



### Login User Screen (5 pts)

By given **email** and **password** the app should be able to login an existing user.

* After a **successful login**:
* a notification message “Login successful.” should be displayed (Bonus\*)
* the user should be **redirected** to the home view.
* In case of **error**:
* an appropriate error message should be displayed (Bonus\*)
* the user should be able to fill the login form again.
* **Form validation** should be the **same** as register.
* Keep the user session data in the browser’s **session/local storage**.
* Clear **all** input fields after **successful** login.

****

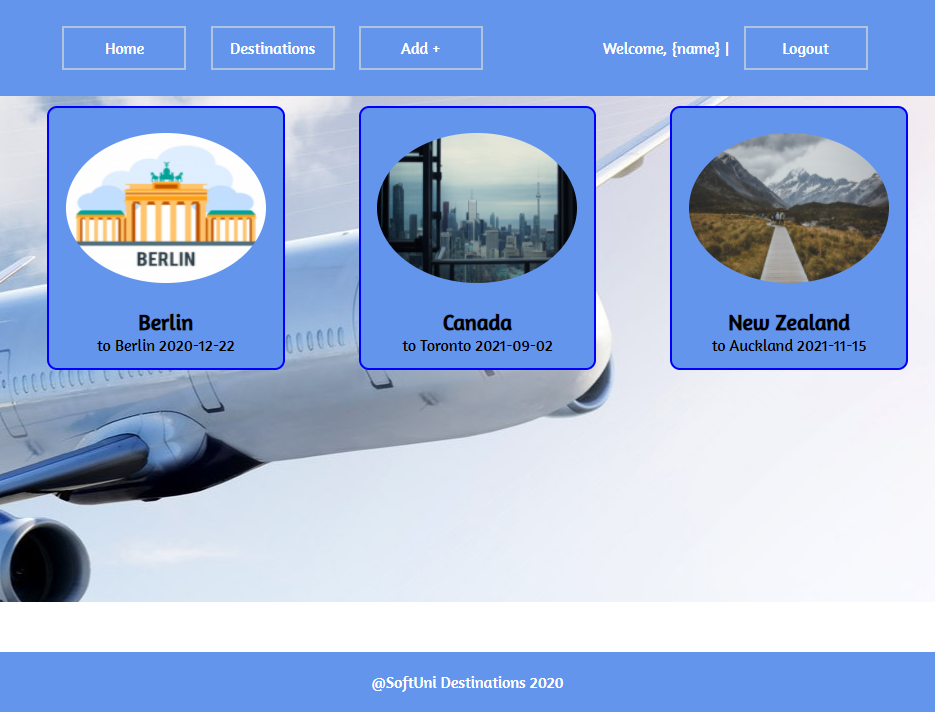
### Logout (5 pts)

Successfully logged in user should be able to **logout** from the app.

* After a **successful** logout:
* a **notification** message “Logout successful.” should be displayed (Bonus\*)
* After successful logout, the **Sign In view** (Login view) should be shown.
* All local information in the browser (**user session data**) about the current user should be deleted.

### Home Screen (List all destinations) (20 points)

When the user is logged in the home screen view shows a list of all users destinations. Should be shown in the following format:

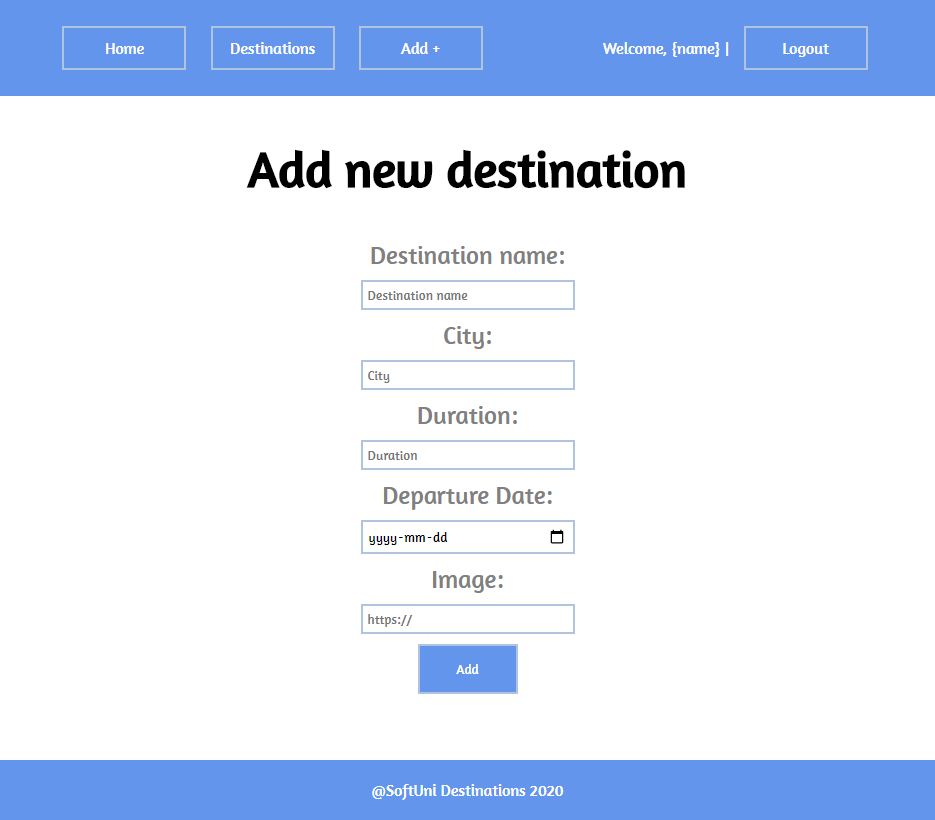


### Add Destination (10 points)

Clicking on **[Add +]** button should **redirect** to form where the user creates a destination.

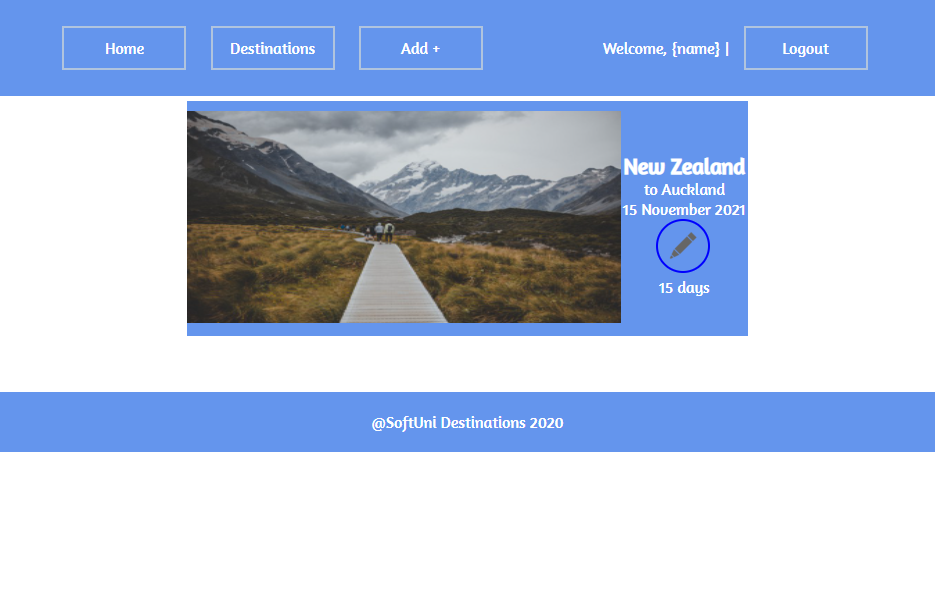
The form should contain the following validations:

* + All **input** fieldsshouldn’t be **empty**.
  + **By** **default**, every newly created destination must have additional information:
    - **Destination name:** string;
    - **City:** string;
    - **Duration:** number (of days) – must be between [1…100];
    - **Departure Date:** string
    - **imgUrl:** string
  + After a **successful** creating Home pageshould be shown.
* The newly added destination should be stored in the database collection "destinations".
* In each case the corresponding **notification** should be shown (Bonus\*).



### Destination Details (10 points)

Clicking on each **individual** destination on the home screen, **redirects** to a destination details page where **additional** information for this destination is shown (destination name, city, departure date, duration).You have to display also the option to edit it (pencil icon).



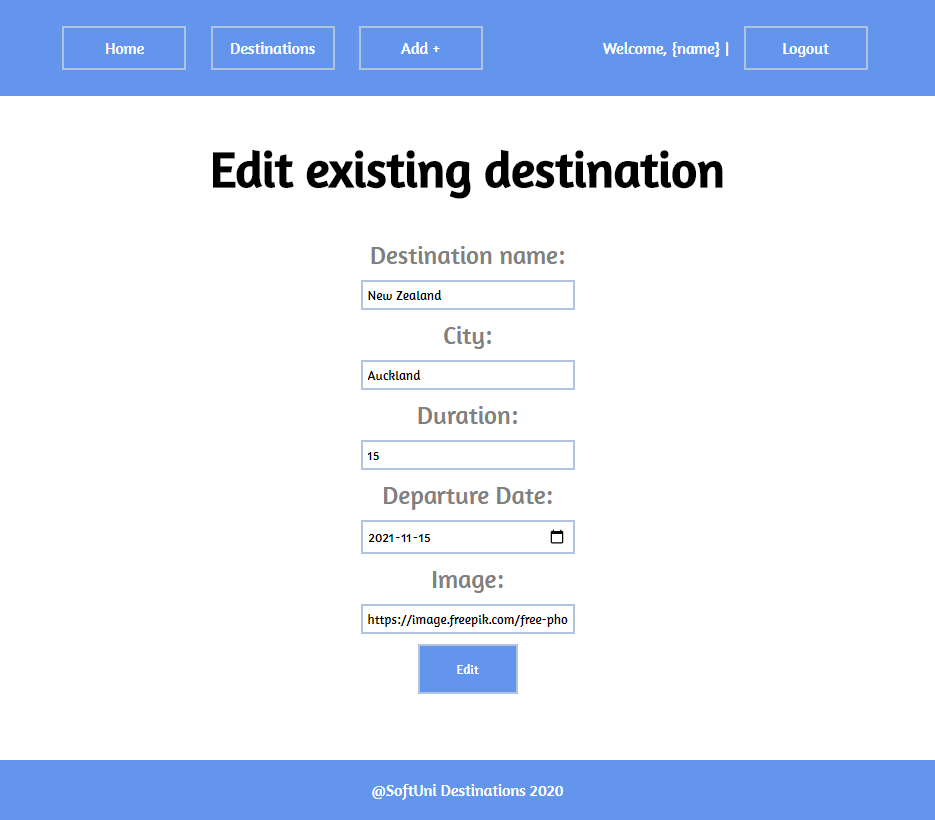
### Edit Destination (10 points)

The user is able to **edit** his destinations.

* Clicking on the edit button **redirects** to a form where the user can **modify** the given destination (all validations in **add a destination** should be followed).
* The form should contain **all validations as the Add Destination** form has.
* After **successfully** editing a destination:

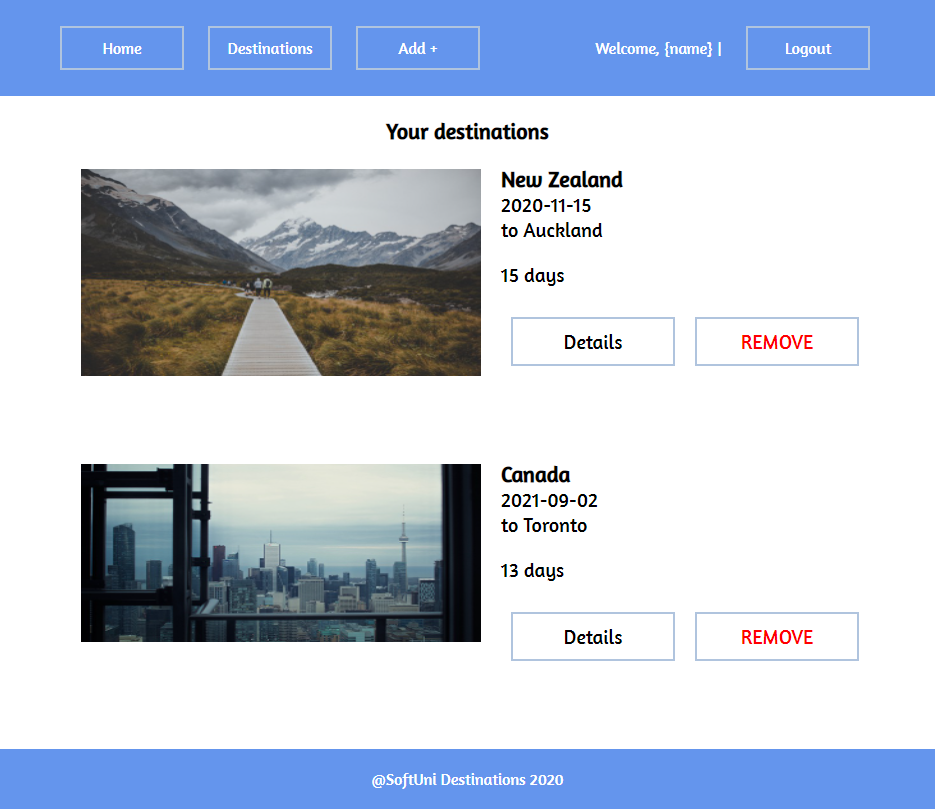
- the user should be **redirected** to the **details** page of the destination

- a message “Successfully edited destination.” should be shown (Bonus\*)

****

### My Destinations (20 points)

All users can view their **own** destinations by clicking on the **[Destinations]** button in the navigation.



### Delete Destination (5 points)

* In the **Destinations** section users can **delete (remove)** their **own** destinations. Deleting is done **instantly**.
* When the user **successfully** deletes a destination:
* the message “Destination deleted.” should be shown (Bonus\*)
* the user should be **redirected** to the **same** page.

### (Bonus): Notifications (10 pts)\*

The application should notify the users about the result of their actions.

* In case of successful action an **informational (green) notification message** should be shown, which disappears automatically after 3 seconds or manually when the user clicks it.



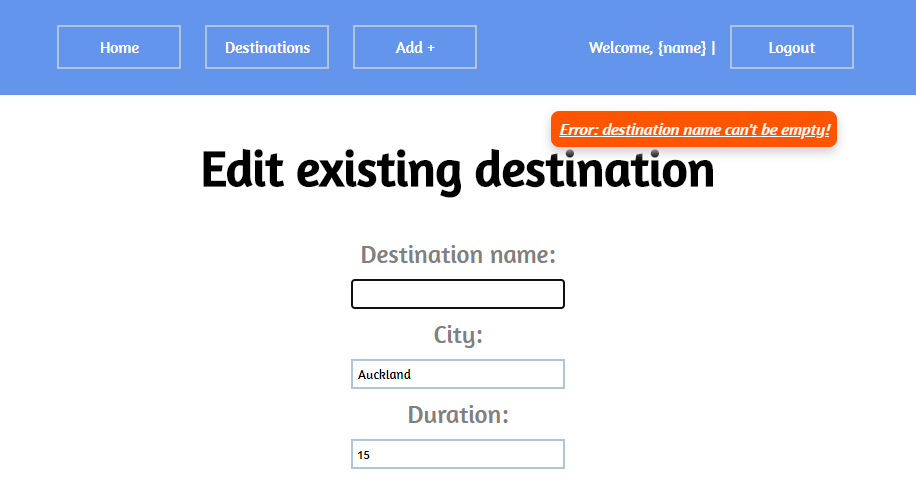
* In case of **error**, an **error notification message (red)** should be shown which disappears on user click.



* During the AJAX calls a **loading notification message (blue)** should be shown. It should disappear automatically as soon as the AJAX call is completed.



Here is an example of notification:

****

## Submitting Your Solution

Exclude the node\_modules folder and ZIP your project. Upload the archive to Judge system.



**Again: zip the project without the** node\_modules **folder!!!**

Refused to execute inline script because it violates the following Content Security Policy directive: "default-src 'self'". Either the 'unsafe-inline' keyword, a hash ('sha256-ktarjbJmNtF8IylbwgjSQoKrcQSdXJkqf60bj4nusHA='), or a nonce ('nonce-...') is required to enable inline execution. Note also that 'script-src' was not explicitly set, so 'default-src' is used as a fallback.