## JS Back-End Exam – BookingUni

## Exam Rules:

1. You have 4 hours
2. When you are ready, delete the ***node\_modules***folder, make sure all dependencies are listed in the ***package.json*** file and submit your archived project at: [SoftUni Judge](https://judge.softuni.bg/Contests/1652/ExpressJS-19-April-2019)
3. You are provided with **HTML & CSS** resources, which you can **modify**
4. Use **Express.js** as a back-end framework
5. Use **MongoDB** as a database with **mongoose**
6. You can use whatever **view engine** you like (Handlebars, EJS, Pug etc.…)

## Application Overview

Get familiar with the provided **HTML & CSS** and create an application for **booking hotel** and **creating offers** for hotels**.**

## Functional Requirements

The Functionality Requirements describe the functionality that the **Application** must support.

**Guest (Not Logged In)**

The **application** should provide Guest (not logged in) users with the functionality to login, register and **view** the Home page.

**Users (Logged In)**

The **application** should provide **Users** (logged in) with the functionality to **view all the hotels which are listed**, **hotel details page of each hotel** and **they should be able to book a hotel**.

**Database Models (10 Pts)**

The **Database** of the BookingUni application needs to support:

### User

* Email - string (required), unique,
* Username – string (required), unique,
* Password - string (required),
* **Booked hotels - a collection of** Hotels the user have booked already,
* **Offered Hotels – a collection of Hotels the user offers**

### Hotel

* **Name - string (required), unique**
* **City - string (required),**
* **Image Url - string (required),**
* **Free Rooms – number (required),** must be between **1 and 100,**
* **Users Booked a room - a collection of Users**
* **Owner – string (required)**

Implement the entities with the **correct datatypes**.

### Application Pages

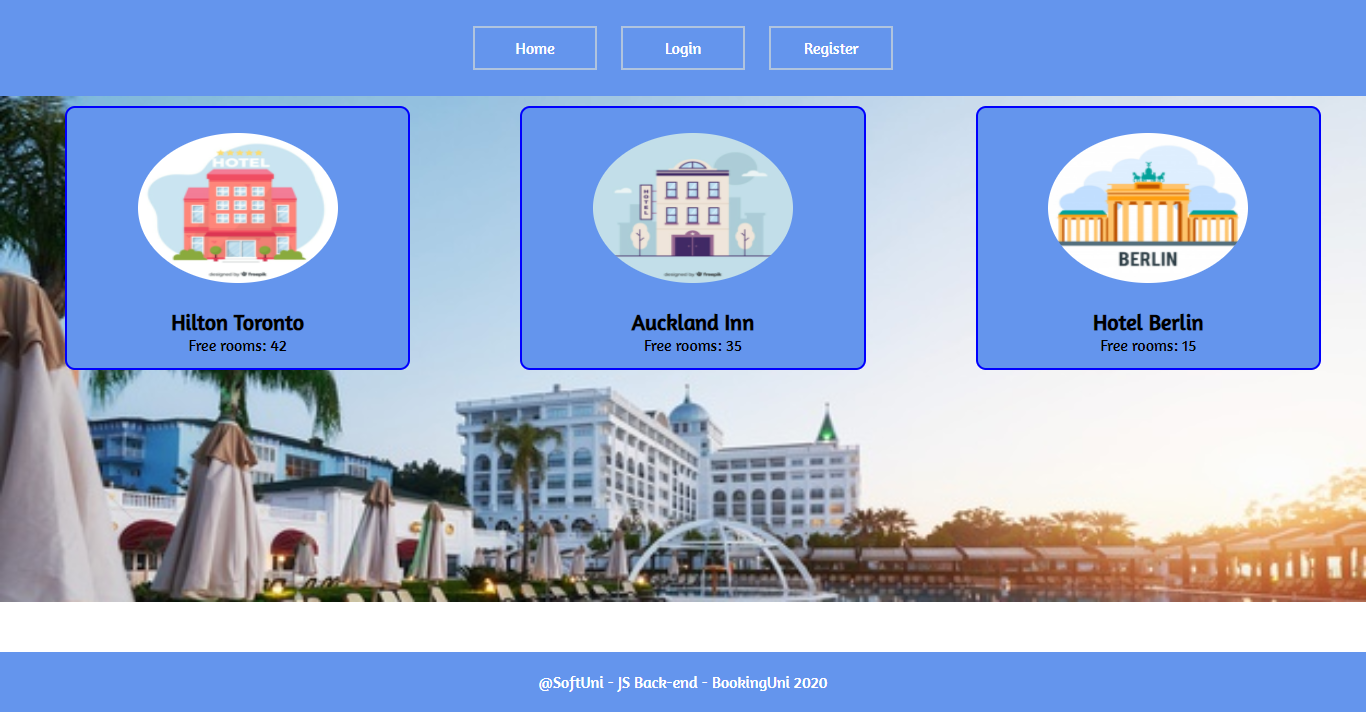
**Guest Pages**

These are the pages and functionalities, accessible by **Guests (logged out users).**

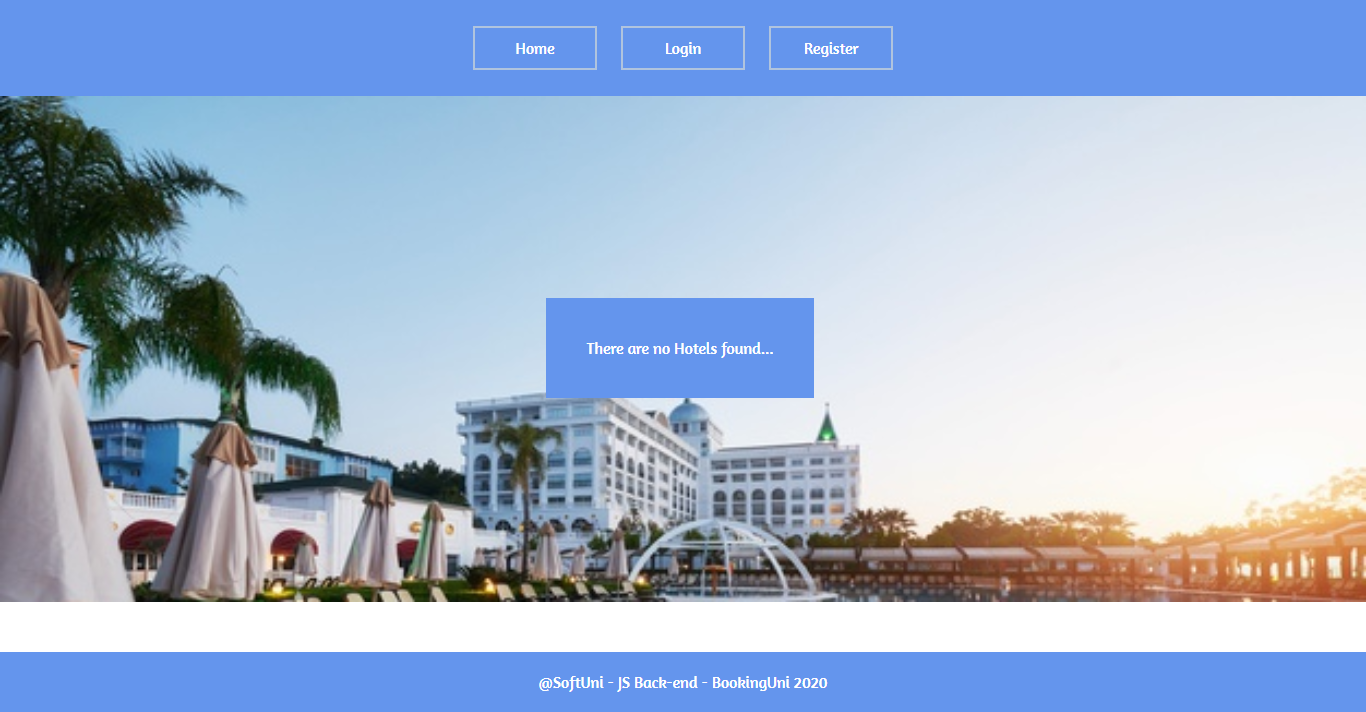
**Home Page (Logged Out User) - (30 Pts)**

List the **hotels ordered** by the **count of free rooms (free rooms are the rooms at the current time, not by hotel creation)** in **descending** order.

If the user is **NOT logged** in by **clicking** on the **hotels picture** should be **redirected to Login** page.

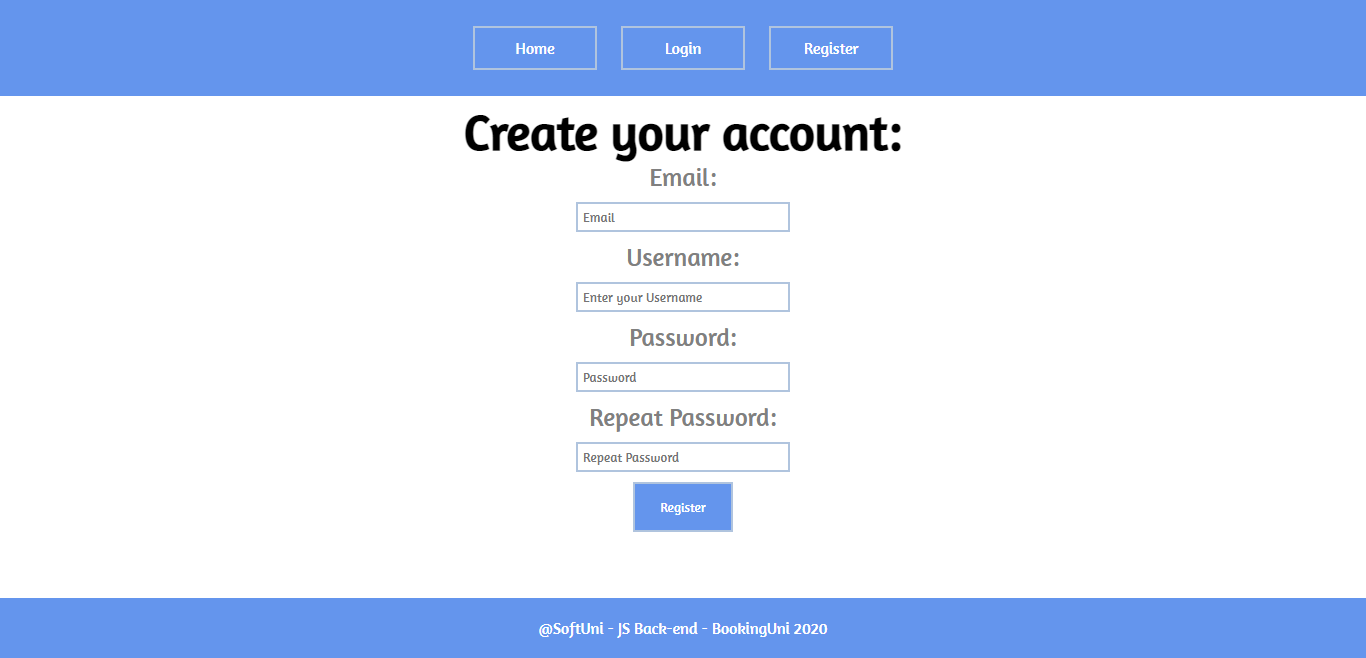


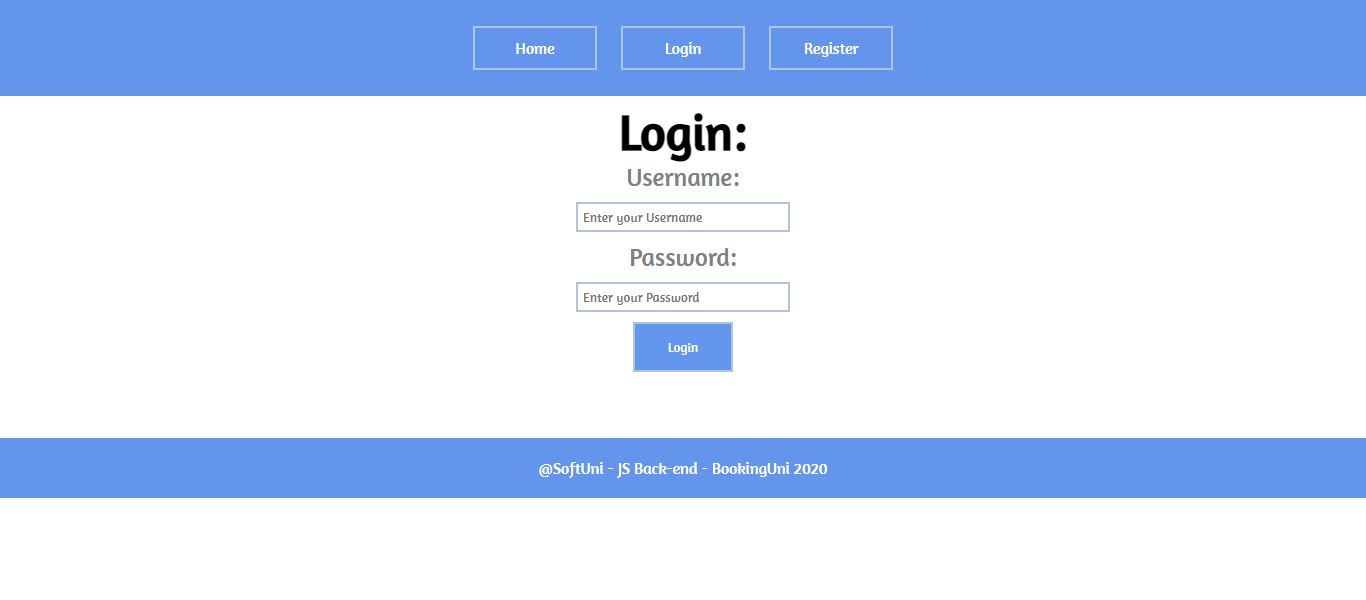
If there are **NO** hotels in the database yet, display **"There are no hotels found…"**



**Register Page (Logged Out User) - (5 Pts)**

Register a user inside the database with **email, username** and **password**. Both **passwords** must **match**! After successful registration, you should **redirect to Home page** and the user should be already **logged in**.





**User Pages**

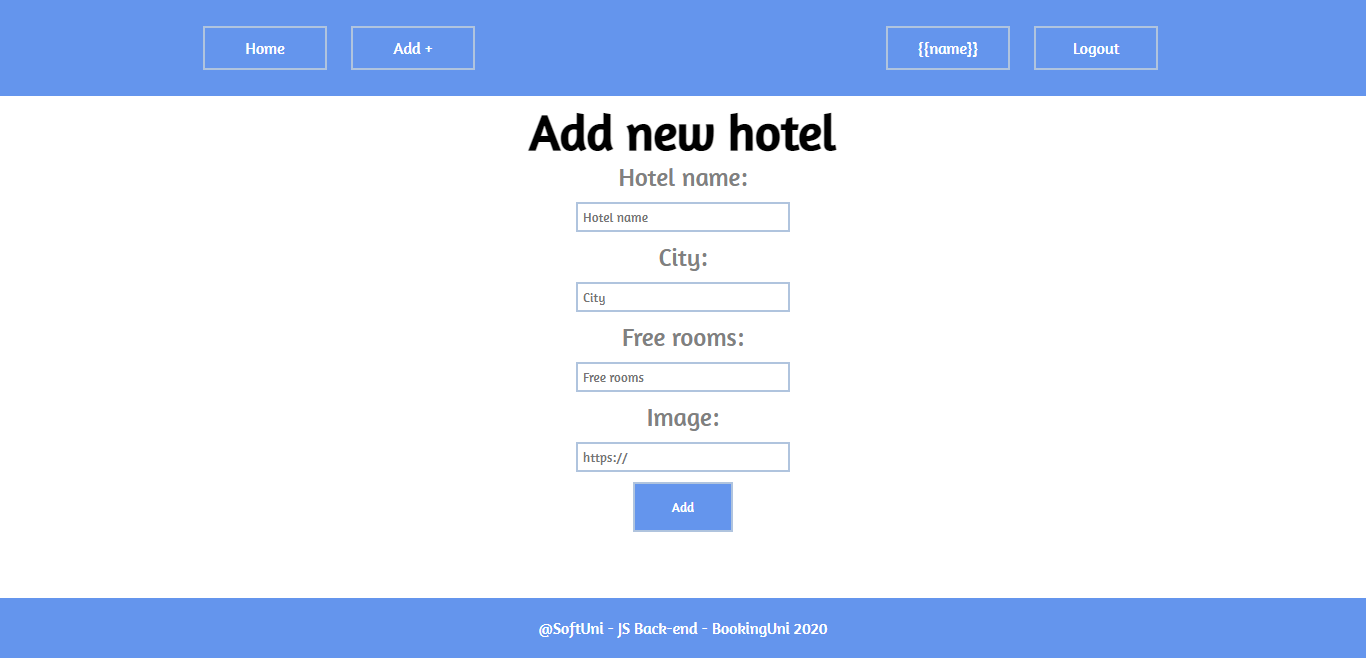
These are the **templates** and **functionalities**, accessible by Users (**logged in** users).

Navigation bar for logged in user:



**Add Hotel Page (Logged in User) - (10 Pts)**

Enter hotel **name**, **city**, **image URL** and **free rooms**. After the hotel is created successfully, you should **redirect to Home page**.

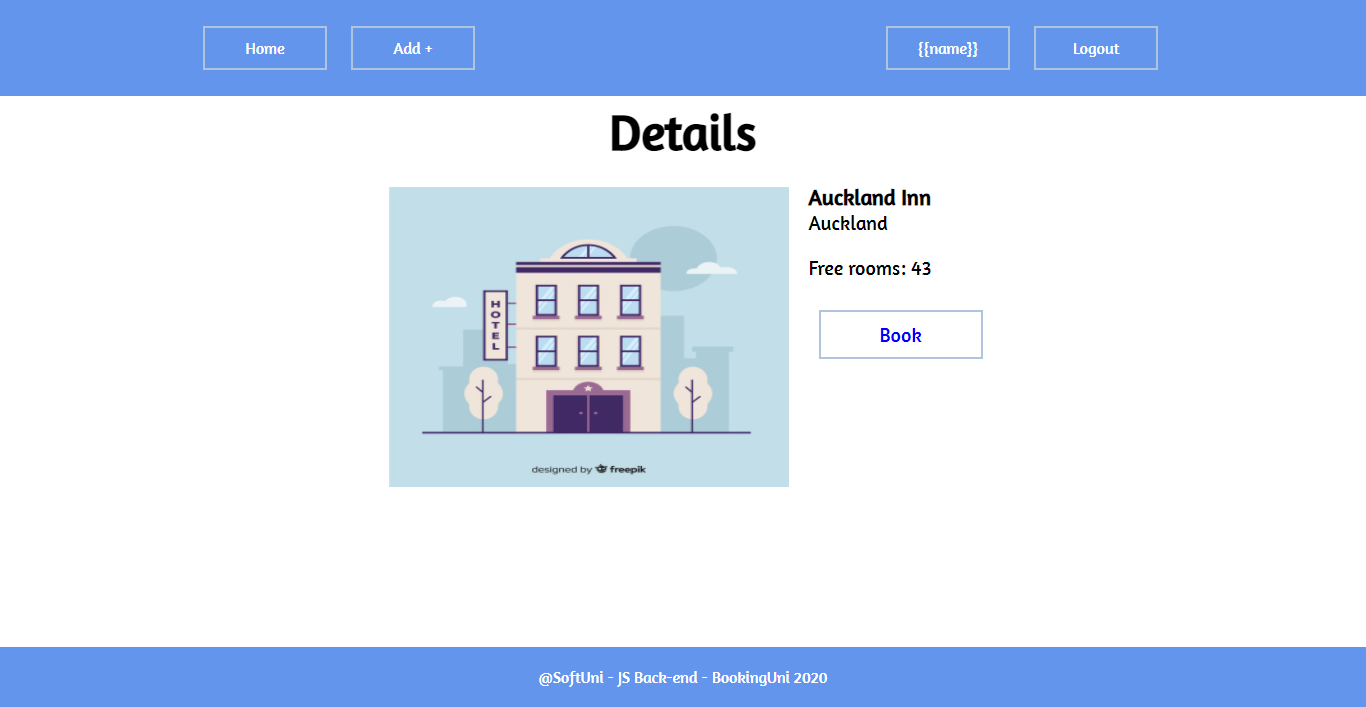


**Logged in Details Page (Logged in User) – (15 Pts)**

**By clicking** the picture of a hotel on **Home page** listed hotels, the user should see the **Details page.**

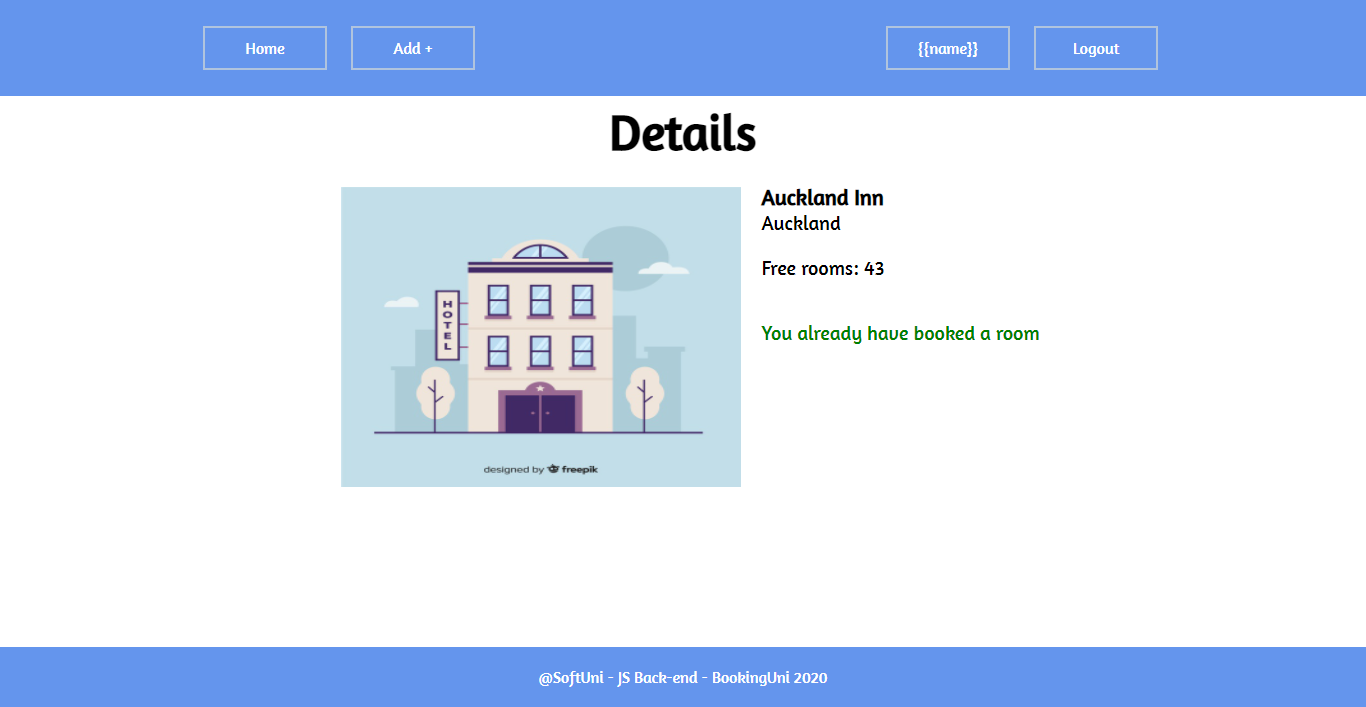
**Not booked hotel (Logged in User)**

If the user **has no reservation for this hotel**, you should display **Book button (a-tag)** andthe user can book a room.

****

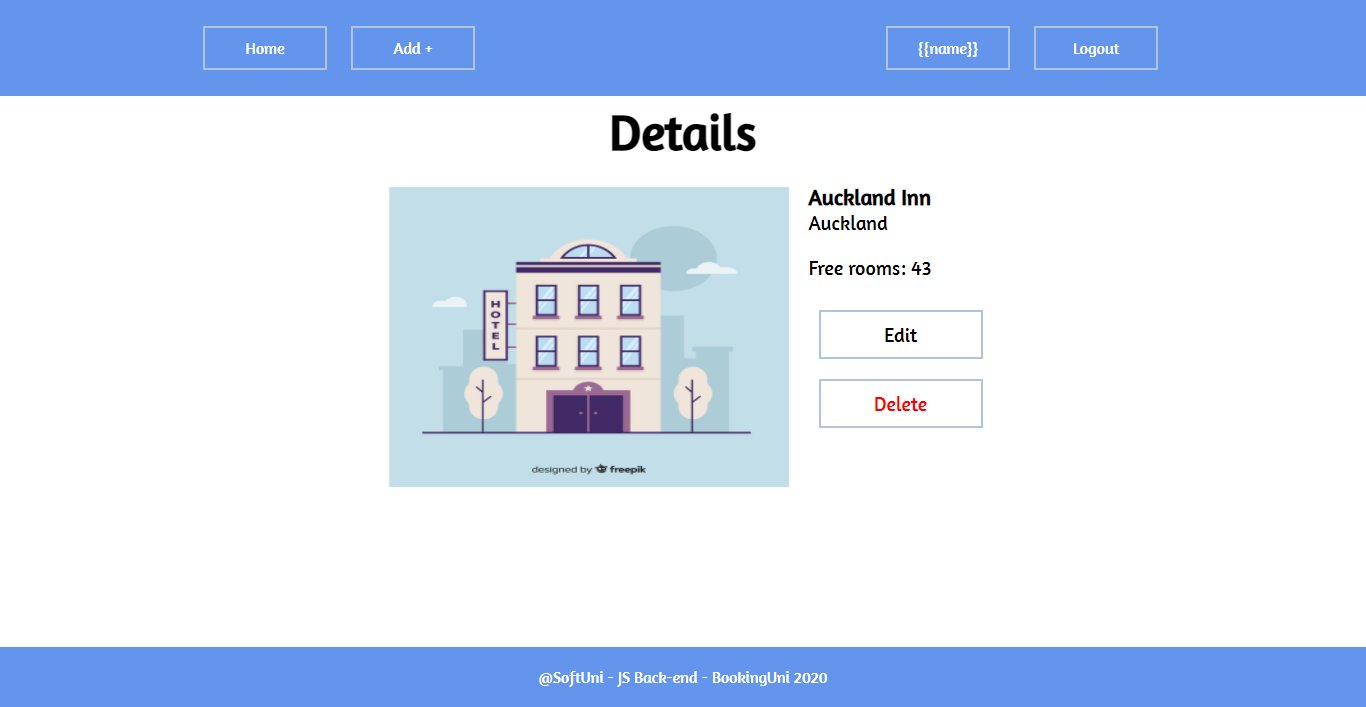
**Booked Hotel (Logged in User)**

**After** the user has booked a room, the **"You already have booking"** should be **shown instead of the [Book] button.**



**Hotel Creator (Logged in User)**

The creator of the hotel should see **[Delete] [Edit]**

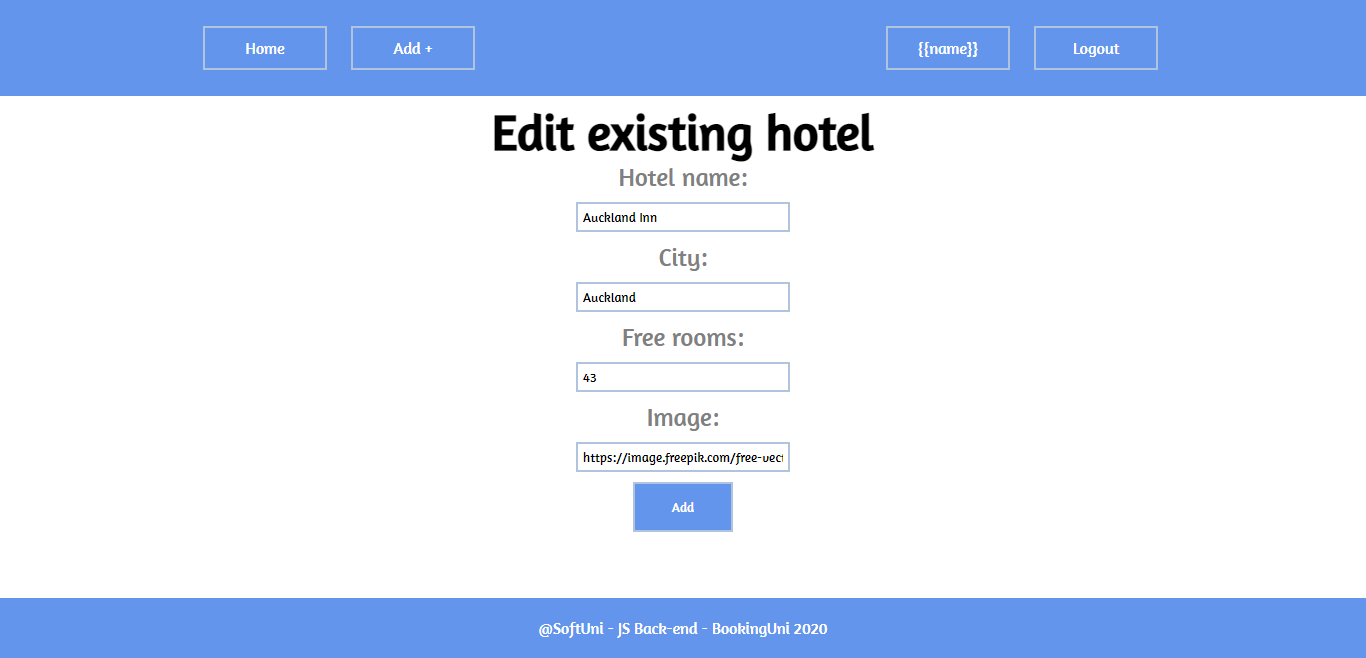


**Delete Hotel (Logged in User) – (5 Pts)**

Upon deleting a lecture (clicking over **[Delete] button**), you should be **redirected to the home page**.

**Edit Hotel (Logged in User) – (10 Pts)**

All form fields should be **filled** with the corresponding information of the selected hotel. When a put request is sent (clicking over **[Edit] button**), you should be **redirected to the Details page**.



**Security Requirements (10 Pts)**

The Security Requirements are mainly **access** requirements. Configurations about which users can access specific functionalities and pages.

* Guest (not logged in) users can access Home page.
* Guest (not logged in) users can access Login page and functionality.
* Guest (not logged in) users can access Register page and functionality.
* Users (logged in) can access Home page page and functionality.
* Users (logged in) can access **Hotel** Details page and functionality.
  + **Users** (not hotel creator) can **Book** **once** a room for hotel.
  + **Users** (hotel creator) can **Edit** and **Delete** the current hotel
* Users (logged in) can access **Create Hotel** page and functionality.
* Users (logged in) can access Logout functionality.

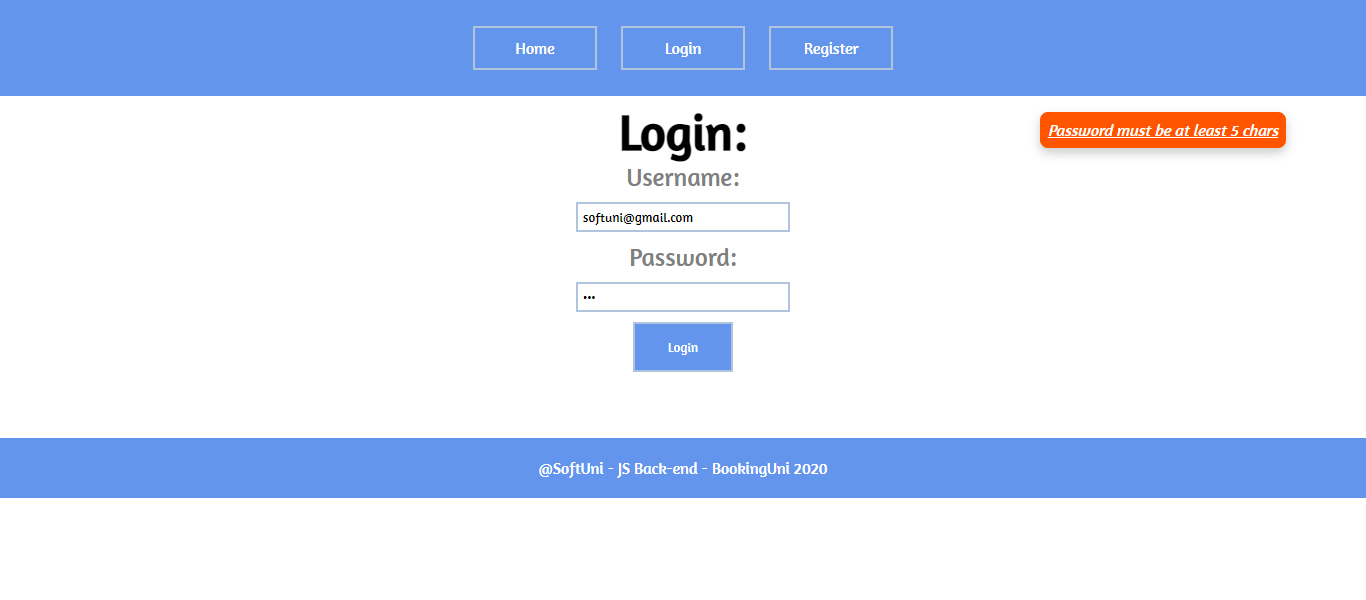
**Validation and Error Handling (10 Pts)**

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

### Login / Register

You should make the following validations:

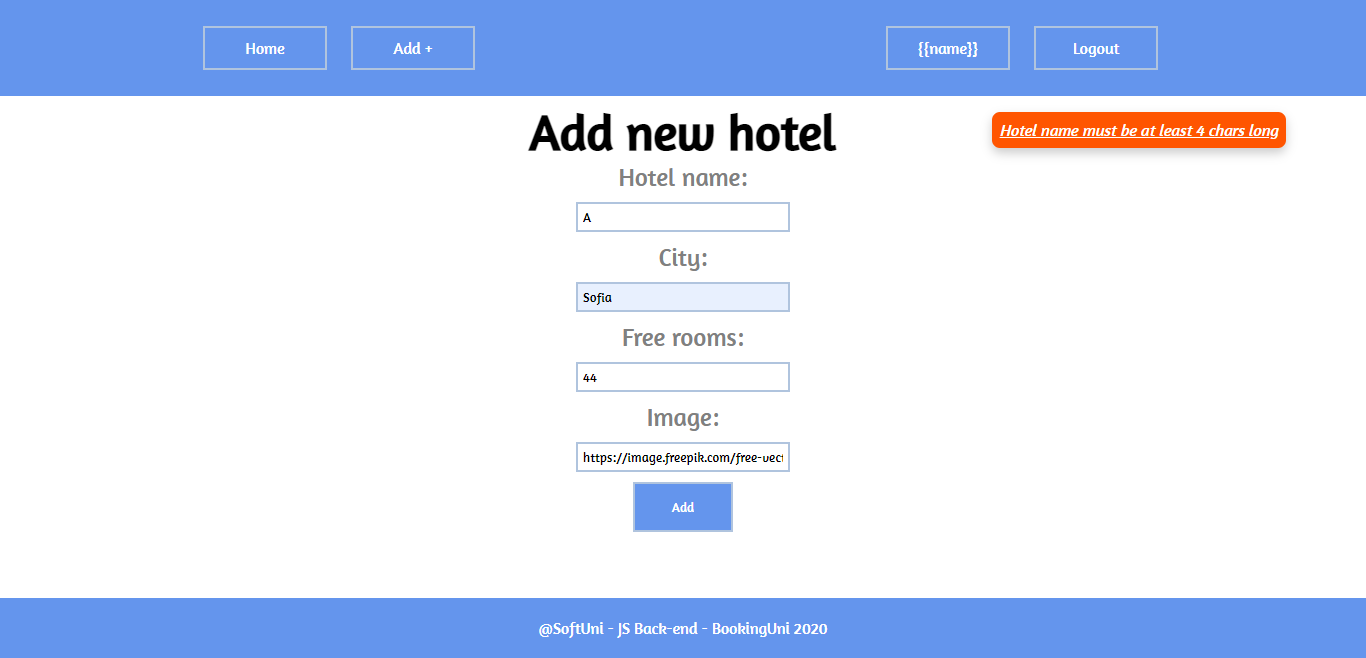
* The **email** should be a valid email and should consist **english** **letters** and **digits**
* The **password** should be **at least 5 characters** long and should consist only **english** **letters** and **digits**
* The **repeat password** should be **equal to the password**



### Hotel

You should make the following validations while **creating** or **editing a** **hotel**:

* The **name** should be **at least 4 characters**
* The **city** should be **at least 3 characters** long
* The **imageUrl** should **starts with http or https**
* The **number** of **free rooms** should be **between 1 and 100**



**\* Bonus (10 Pts) – Profile Page**

**Profile Page** shows the information of the user as shown on the picture.

**Reservations** – is a string: **all the names** of the **booked from the user** **hotels** separated by semi-colon.

