INFO3180 Project 1 (30 marks)

Due: March 12, 2023 at 11:55 PM

At the end of this project you will have a Flask based application that can accept and display information on properties available for rent/sale. The property information will also be stored in a PostgreSQL database.

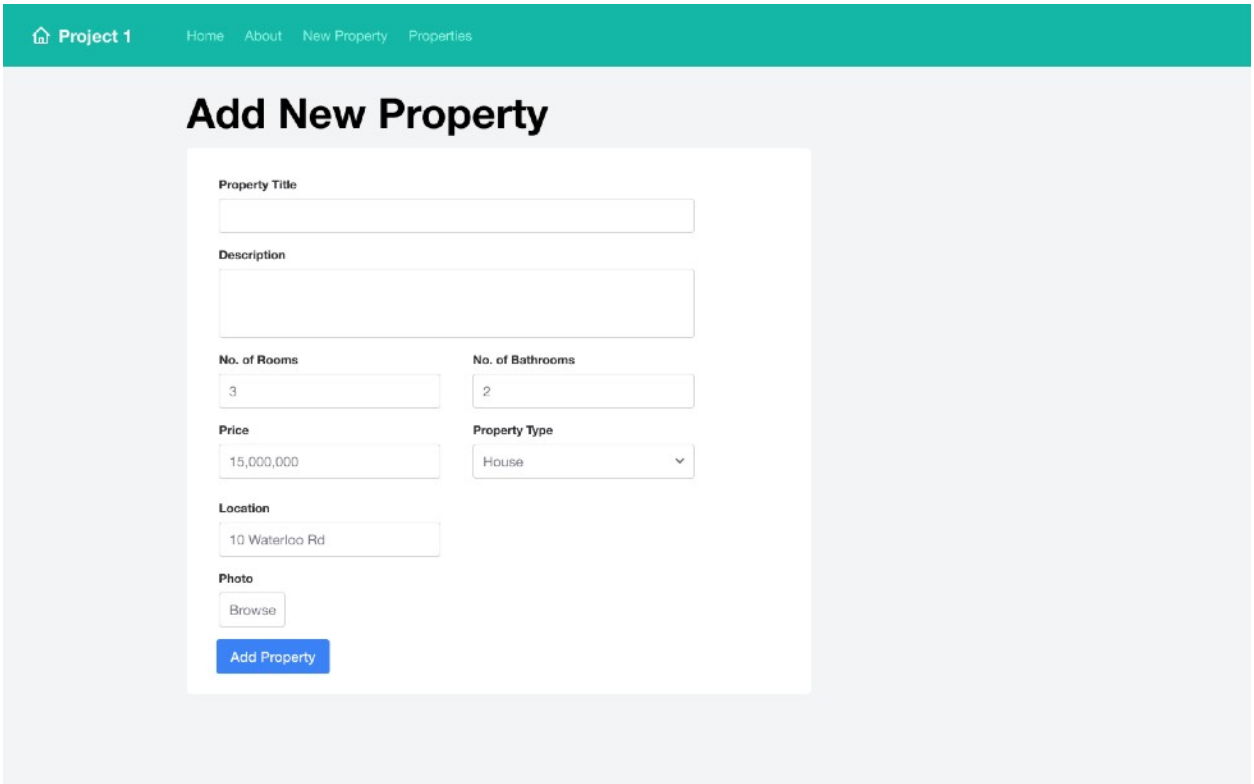

Figure 1: Add New Property Form

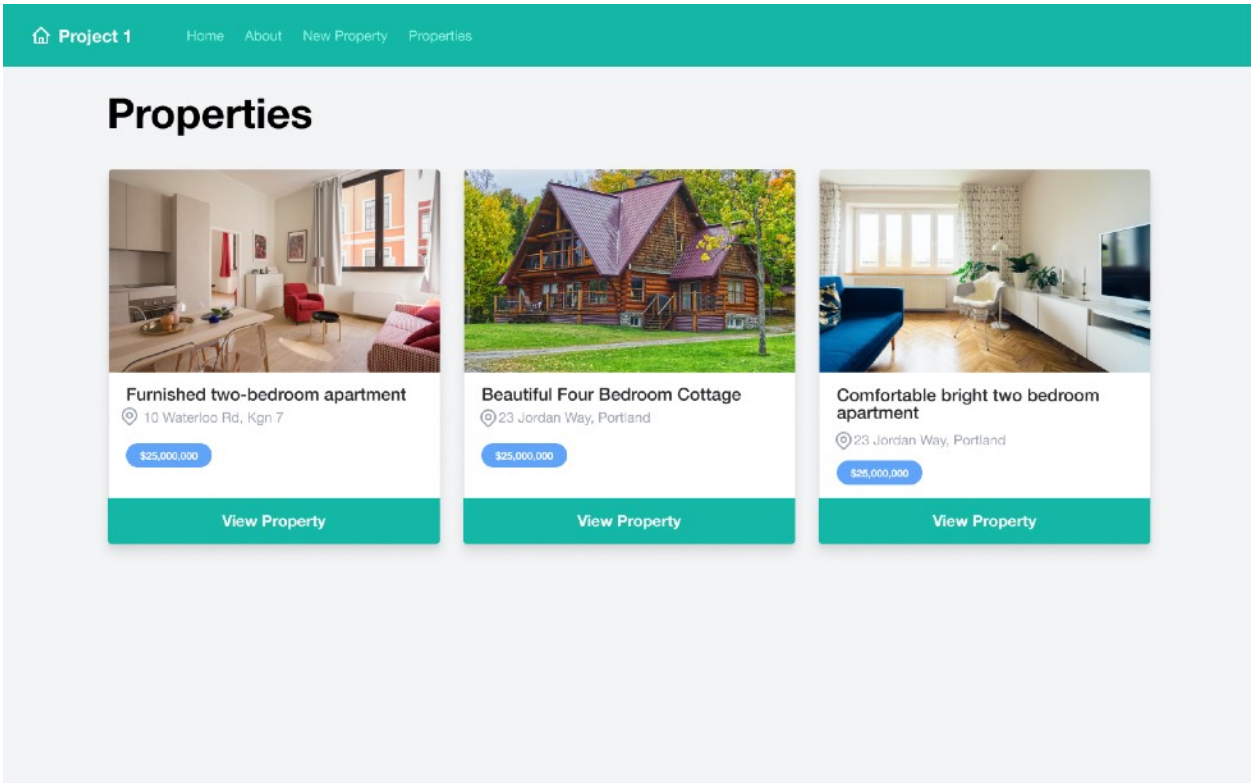

Figure 2: List of Properties available

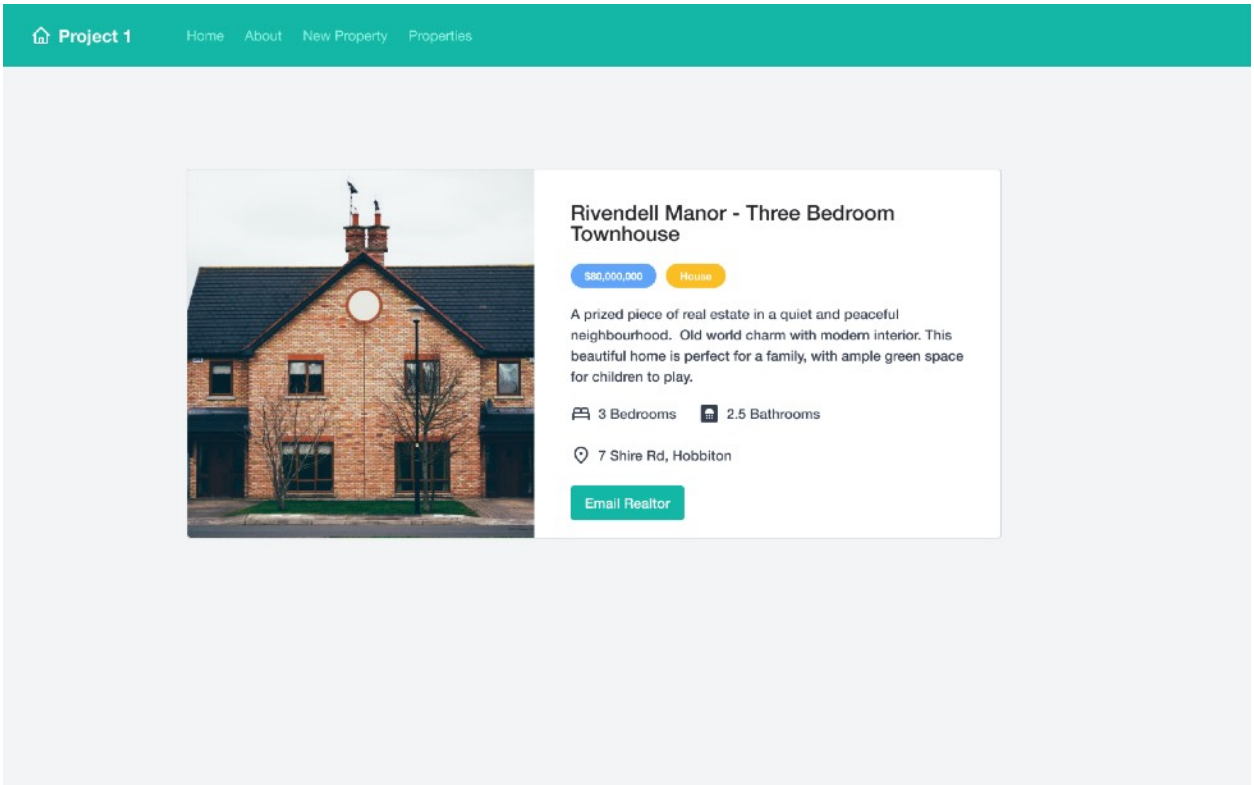

Figure 3: Individual Property Information

Specifications

Use the knowledge you have gained in your Lectures, Tutorials and Labs to create a Flask App that accepts input for a user to create a new property and also display both a list of those properties as well as an individual property.

Feel free to use the Flask Starter code at https://github.com/uwi-info3180/flask_starter as a basis to start your application. Remember though you will also need to update your **`requirements.txt`** file to include any additional libraries you need for your project.

The following routes will need to be created and appropriate templates rendered:

1. "**`/properties/create`**" For displaying the form to add a new property. (*See Figure 1*)
2. "**`/properties`**" For displaying a list of all properties in the database. (*See Figure 2*)
3. "**`/properties/<propertyid>`**" For viewing an individual property by the specific property id. (S*ee Figure 3*)

The add new property form must be created using Flask-WTF and contain the following fields:

1. Text fields for **`title`**, **`number of bedrooms`**, **`number of bathrooms`**, **`location`** and **`price`**.
2. Select (option) field for **`type`** (whether House or Apartment)
3. Textarea field for a short **`description`**.
4. File upload field called **`photo`** which accepts the image of the Property.

Upon submission, the form should make a **POST** request and validate the user input to prevent bad data. A unique **`id`** should be generated (e.g. an auto incrementing id field in your model) and also the *filename* of the **photo** for the new property should be saved in the database. All of this input must be stored in a PostgreSQL database.

Once a property is successfully added the user should be redirected to the "**`/properties`**" route and a flash message should be displayed notifying the user that the property was successfully added.

Note: You **MUST** generate a migration file for your *`Property`* model so that the database can be recreated. Remember to ensure your migration file is committed to your repository.

For the list of properties page, you are to display a list of *all* properties. Each property should display the *photo*, *title*, *location* , price and *a button/ link* that when clicked should carry you to the individual property page where you can view more details. (See Figure 2).

On the Individual property page, you should have the property's *photo*, *title*, description, *no of bedrooms, no of bathrooms*, *location* and *price*, *along with whether or not it is a house or apartment*. You should also have a *button* to "Email Realtor" (See Figure 3). Please note the button does not need to do anything for this project.

Note: You must also add navigation links to the "**`/property`**" and "**`/properties`**" routes to your **`header.html`** file.

Submission

Submit your code via the "**Project 1 Submission**" link on OurVLE. You should submit the following links:

1. Your Github repository URL for your Flask app e.g. **https://github.com/{yourusername}/info3180-project1**

Grading

1. 3 routes `/properties/create`, `/properties`, `/properties/<propertyid>` should be defined along with their respective view functions. (5 marks)
2. `/properties/create` has form with *title*, *description*, *no of bedrooms*, *no of bathrooms*, *location*, file upload field called *photo* and select *property type* field. (4 marks)
3. The Form successfully submits and property added to database. A flash message should also be displayed to give the user feedback for the successful submission. (2 marks)
4. File successfully uploads and filename stored in database. (2 marks)
5. You should be able to view the specific user property at `/properties/<propertyid>` (5 marks)
6. You should be able to view a list of properties added at `/properties` (3 marks)
7. A Property database model should exist in `models.py`. (2 marks)
8. Database Migration(s) should be created and the database should be able to be recreated from that migration. (2 marks)
9. Navigation links for "New Property" and "Properties" are in the navigation bar at the top. (2 marks)
10. 'Property', 'Properties' and 'Add Property Form' should look similar to screenshots. (3 marks)