ENSF-381: Full Stack Web Development Laboratory

Ahmad Abdellatif and Novarun Deb
Department of Electrical & Software Engineering
University of Calgary
Lab 9

Objectives

Welcome to the ENSF381 course lab! In this lab, we will build a full-stack web application that allows users to log in and predict house rental prices based on input features. will create a React frontend with routing, styled forms, and API integration, and a Flask backend that handles user authentication and interacts with a machine learning model to generate predictions.

Groups

Lab instructions must be followed in groups **of two students**.

Submission

You must submit the complete source code, ensuring it can be executed without any modifications. Also, if requested by the instructor, you may need to submit the corresponding documentation file (e.g., word and image). Only one member of the group needs to submit the assignment, but the submission must include the names and UCIDs of all group members at the top of the code.

Deadline

Lab exercises must be submitted by **11:55 PM on the same day as the lab session**. Submissions made within 24 hours after the deadline will receive a maximum of 50% of the mark. Submissions made beyond 24 hours will not be evaluated and will receive a grade of zero.

Academic Misconduct

Academic Misconduct refers to student behavior which compromises proper assessment of a student's academic activities and includes: cheating; fabrication; falsification; plagiarism; unauthorized assistance; failure to comply with an instructor's expectations regarding conduct required of students completing academic assessments in their courses; and failure to comply with exam regulations applied by the Registrar.

For more information on the University of Calgary Student Academic Misconduct Policy and Procedure and the SSE Academic Misconduct Operating Standard,

please visit: https://schulich.ucalgary.ca/current-students/undergraduate/student-resources/policies-and-procedures

Frontend: React Application

The frontend has two main pages, each built as a React component:

1. `Login`: For authenticating users.
2. `HousePricePredictor`: For entering house details and viewing the estimated rent.

Login Page

- Create a login form with two input fields: one for the username and one for the password.
- Both fields must be required.
- When the form is submitted, send a `POST` request to the login API endpoint (`/validate_login`) on the backend.
- If the login is successful, navigate to the house price predictor page (`/predict` route).
- If the login fails, show a clear and visible error message.
- Next, we need to style the page so it is clean and user-friendly. Apply the following styles:
  - Wrap the entire form in a container that is **centered horizontally** on the page.
  - The container should have a **maximum width of 400 pixels**.
  - Add **padding** (`1rem`) inside the container to give space around the content.
  - Add a **light border** and **rounded corners** to the container.
  - Add a **subtle shadow** to give a card-like appearance.
  - Each form field (username and password) should be placed **under a label** and arranged vertically.
  - The input fields should and include some padding inside the fields (e.g., `0.5rem`).
  - Add **spacing (**`margin-bottom: 1rem`**) between each field** so the form looks clean and organized.
  - Style the submit button with a **blue background** (`#007BFF`), **white text**, and **rounded corners**. The button should have enough padding (`0.5rem 1rem`) to make it easy to click.
  - When hovered, the button should show a slightly **darker blue** to indicate interactivity.
  - If there is an error (like invalid login), display the message in **red** below the form.

House Price Predictor Page

- Create a form that allows the user to input various house-related details. Make sure the following fields are included in your form (all fields must be **required**, except for the checkbox which should default to unchecked):

  - City
  - Province
  - Latitude
  - Longitude
  - Lease Term

    - Type of House
    - Number of Beds
    - Number of Baths
    - Square Feet
    - Furnishing (should be a dropdown with three options: Unfurnished, Partially Furnished, Fully Furnished)
    - Smoking (Yes/No or similar)
    - Pets (a checkbox to indicate whether the user has pets)
- Make sure the **keys in your request JSON** (coming from the frontend) **match exactly** the ones expected by the backend. The backend expects these exact keys:

```
'city', 'province', 'latitude', 'longitude',
'lease_term', 'type', 'beds', 'baths', 'sq_feet',
'furnishing', 'smoking', 'pets'
```

- After the form is submitted, send the data to the prediction API endpoint (`/predict_house_price`) on the backend.
- Display the **predicted rental price** below the form once a response is received.
- Next, we need to style the page so it is clean and user-friendly. Apply the following styles:

    - Use a **container** with a **maximum width of 600 pixels** with 24px padding on all sides and **center it horizontally** on the page.
    - Center the main container horizontally using auto margins, and add 40px of top margin.
    - Apply a 1px solid light gray border, 8px rounded corners, and a soft shadow to create a card-style appearance.
    - Arrange all form fields in a vertical column with 12px spacing between them.
    - Each field should include a label above the input or dropdown, and be full-width (100%) with 8px internal padding.
    - Inputs and dropdowns should have a 1px light gray border and 4px rounded corners.
    - The “Furnishing” field should be a dropdown with three options: Unfurnished, Partially Furnished, and Fully Furnished.
    - The “Pets” field should be a checkbox with a label, spaced like the other fields.
    - The submit button should have a blue background (#007BFF), white text, 12px vertical and 24px horizontal padding, 4px rounded corners, and change to a pointer cursor on hover.
    - On hover, the button background should darken slightly to show interactivity.
    - Display the prediction result in a box below the form, with 16px top margin, light green background (#DFF0D8), 1px solid green border (#3C763D), 12px padding, 4px rounded corners, and bold text.

App Component

In the `App.js` file, you are responsible for setting up the routing between the main components of your React application. Use `react-router-dom` to define the routes. Note that `App.js` should focus solely on routing and not contain any UI elements or forms.

Backend: Flask Application

You will write a Flask server that supports two API endpoints. Create a Flask app with two POST endpoints: `/validate_login` and `/predict_house_price`. Next, enable Cross-Origin Resource Sharing (CORS) so your frontend can connect to the backend during development.

Validate Login API

- For the `/validate_login` endpoint, accept a JSON request containing `username` and `password`.
- Use the following dictionary of predefined user credentials to check if the username exists and the password matches:

```
alice: password123
  bob: secure456
charlie: qwerty789
  diana: hunter2
   eve: passpass
  frank: letmein
 grace: trustno1
 heidi: admin123
  ivan: welcome1
 judy: password1
```

- If credentials are valid, return a JSON response with `"success": true` and a success message.
- If credentials are invalid, return `"success": false` and an appropriate error message.

Predict House Price API

This API uses a **Random Forest Regressor**—a powerful ensemble machine learning model—for predicting house rental prices based on property and location features. It takes different features as an input (e.g., city, and square footage) and predicts rent price (continuous value) as an output.

- Place `random_forest_model.pkl` model file into a folder named `src` (you will find the model file in the supplementary material provided with the lab).

- For the /`predict_house_price` endpoint, accept a JSON request containing all required house data fields from the form. Paste this in your Flask file:

```
model = joblib.load("./src/random_forest_model.pkl")

data = request.json

cats = True if 'pets' in data and data['pets'] else False
dogs = True if 'pets' in data and data['pets'] else False

sample_data = [
      data['city'],
      data['province'],
      float(data['latitude']),
      float(data['longitude']),
      data['lease_term'],
      data['type'],
      float(data['beds']),
      float(data['baths']),
      float(data['sq_feet']),
      data['furnishing'],
      data['smoking'],
      cats,
      dogs
]

sample_df = pd.DataFrame([sample_data], columns=[
      'city', 'province', 'latitude', 'longitude', 'lease_term',
      'type', 'beds', 'baths', 'sq_feet', 'furnishing',
      'smoking', 'cats', 'dogs'
])

predicted_price = model.predict(sample_df)

return jsonify({"predicted_price": float(predicted_price[0])})
```

Submission:

Fill out the names and UCIDs of all group members in Answer_sheet. Also, upload both the completed Answer_sheet.docx and the code to D2L.