Hackathon Report:
Australian Cities Rainfall Prediction

Project Overview
This project aims to predict the rainfall in an Australian city for the next day, using machine learning techniques and weather data obtained from the OpenWeather API. The team consists of five members, with three first-year students focusing on frontend and backend development, and two second-year students focusing on machine learning aspects of the project.

Team Composition
- **First Year**
    1. Deepak Kumar (Frontend/Backend Developer)
    2. Hemant Meena (Frontend/Backend Developer)
    3. Akash Deep (Frontend/Backend Developer)

- **Second Year**
    1. Narayan Jat (Machine Learning Specialist)
    2. Pankaj Yadav (Machine Learning Specialist)

Project Details
The goal of this project is to create a simple application where users can input an Australian city name, and the system will predict the rainfall for the following day. The application utilizes the OpenWeather API to obtain real-time weather information, which serves as input for the machine learning model.

Frontend/Backend Development
Deepak Kumar, Hemant Meena, and Akash Deep worked on the frontend and backend parts of the project. They designed the user interface, created the form to accept user input, and implemented the logic to interact with the OpenWeather API. The frontend was developed using modern web technologies (such as HTML, CSS, and JavaScript), while the backend utilized a server framework (Flask) to handle API requests.

Machine Learning Development
Narayan Jat and Pankaj Yadav were responsible for developing the machine learning model used to predict rainfall. They used a pre-existing dataset, including temperature, humidity, wind speed, pressure, and other relevant factors.. The model was trained on historical weather data to understand the patterns and correlations between different parameters and rainfall.

The machine learning model was developed using popular frameworks Scikit-learn. It underwent a series of training and testing phases to improve its accuracy and reliability.

Workflow

1. Model Training: The machine learning team trained the model using the pre-existing data, optimizing it to achieve the best possible accuracy.

2. Frontend and Backend Development: The frontend team designed a user-friendly interface, while the backend team developed the server-side logic to process API & form requests, return predictions.

3. Integration and Testing: The frontend, backend, and machine learning components were integrated into a cohesive application. The team conducted extensive testing to ensure accurate predictions and a seamless user experience.

Libraries Used:

Flask: Flask is a lightweight WSGI web application framework in Python. It's easy to use and provides tools, libraries, and patterns for building web applications.

Requests: Requests is an elegant and simple HTTP library for Python, used for making HTTP requests. It allows sending HTTP requests and receiving HTTP responses from web servers. In this application, it's used to make requests to the OpenWeatherMap API to fetch weather data based on the user's input.

Algorithm overview:

1. **Home Page Rendering:** When a user accesses the root URL ('/'), the home() function is called, which renders the 'index.html' template using the render_template() function. This is the landing page where users can input the city for which they want to check the weather.

2. **Input Validation:** The function first checks if the request method is POST and attempts to retrieve the city name from the form data. If the city field is empty, it returns a JSON response with an error message prompting the user to enter both the city and country.

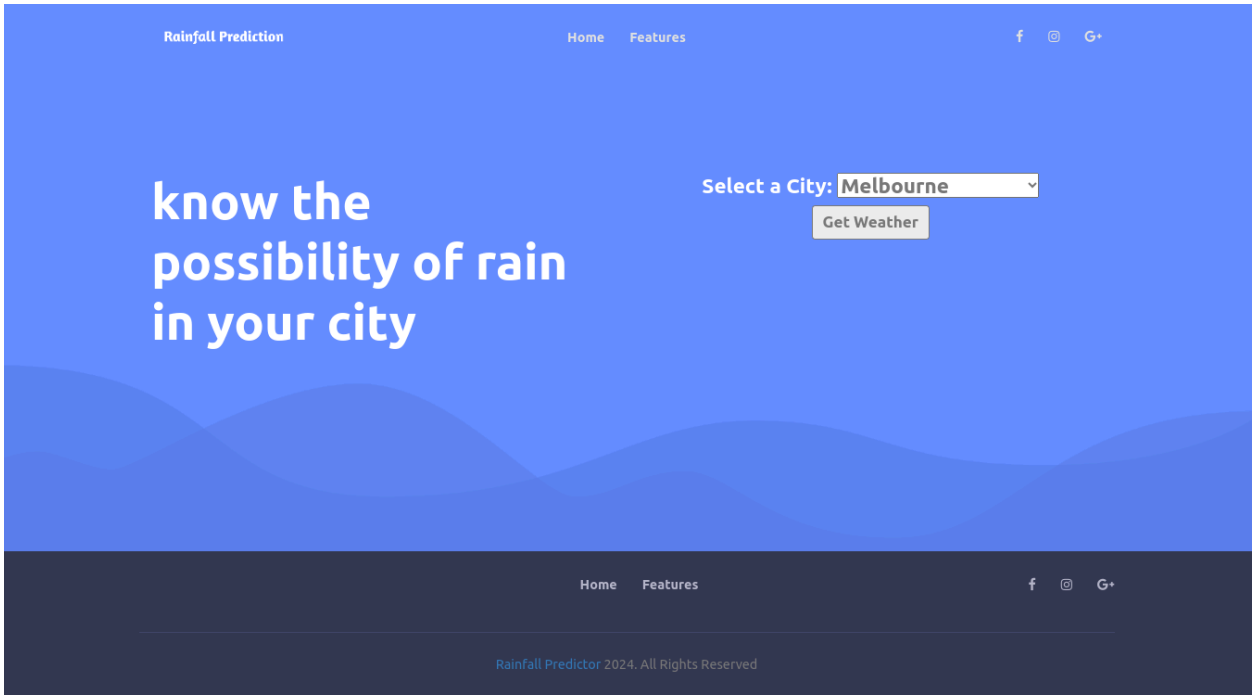
3. **API Request:** If the city name is provided, the function constructs a URL for the OpenWeatherMap API with the city name and sends a GET request using the requests.get() function. The API response is then parsed as JSON.

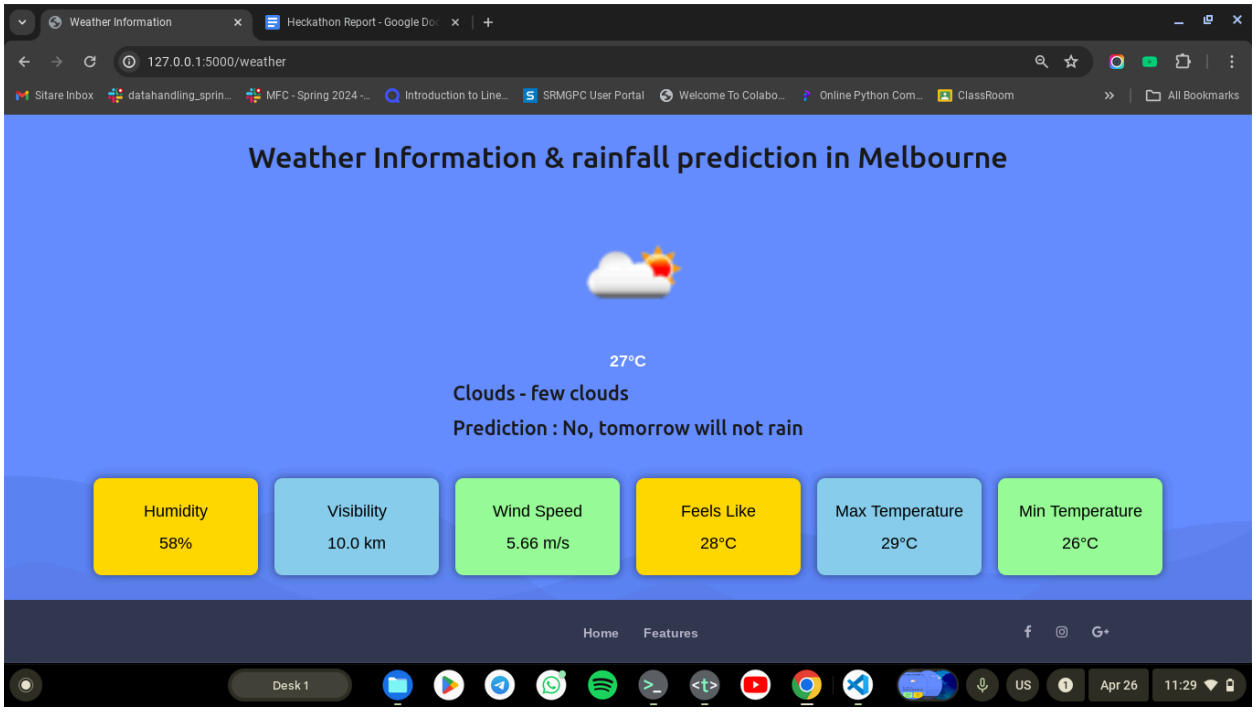
4. **Data Processing:** If the API request is successful (status code 200), the weather data received is processed. The temperature is converted from Kelvin to Celsius, and various weather details such as temperature, feels like temperature, humidity, etc., are extracted from the JSON response.

5. **Template Rendering:** The weather details are passed to the 'weather.html' template using the render_template() function. This template is responsible for displaying the weather information to the user.

6. **Error Handling:** If there is an error during any step of the process (e.g., invalid city name, API request failure), appropriate error messages are displayed to the user either as a JSON response or by rendering the 'home.html' template with an error message.

Test Cases:





Contributions:

Hemant :

- Integrated joblib for loading pre-trained ML models in Flask
- Utilized pandas for efficient data manipulation in the backend
- Optimized frontend performance

Deepak :

- Implemented dynamic rendering of weather data on the frontend.
- Implemented weather data retrieval and input validation in Flask.
- Error handling
- API handling for OpenWeather

Akash Deep:

- Managed routing, rendering, and logic in Flask.
- Designed and developed UI elements using HTML/CSS
- Handled frontend integration with Flask backend.

Challenges and Solutions

- **Data Variability:** The weather data is inherently variable, leading to challenges in training a consistent model. The team tackled this by incorporating a large dataset and applying data normalization techniques.

- **Model Accuracy:** Ensuring the model's accuracy was a challenge due to the complex nature of weather prediction. The team addressed this by fine-tuning the model and using validation techniques.

Conclusion and Future Work

The project successfully created a user-friendly application that predicts rainfall for Australian cities based on input from the OpenWeather API and a trained machine learning model. The application is capable of providing accurate predictions, with the team ensuring robustness and scalability.

For future work, the team plans to enhance the model's accuracy further, explore additional weather parameters, and expand the application's capabilities to include more cities and weather conditions. They also aim to incorporate real-time weather updates and user feedback to improve the prediction system over time.