Weather Application

Group Members:

- Ryan John A. Villarete
- Kyrl T. Tulio
- Joshua M. Padilla

1. Application Design

Our Weather Application is designed to provide real-time weather information to users based on their location. The application utilizes OpenWeatherMap's APIs for current weather data, hourly historical data, and geolocation. Users can input a city name or use geolocation to view the weather and additional details such as temperature, humidity, wind speed, and a weather icon representing the current weather conditions.

Frontend Design: The user interface (UI) of the application is divided into two main panels:

- **Left Panel:** Displays the weather information (temperature, humidity, wind speed, etc.).
- **Right Panel:** Shows a map of the current location using the Leaflet API, which provides interactive maps.

The UI is responsive, and the weather data updates dynamically based on user input or geolocation.

2. API Integration

We integrated the following APIs into our application:

- **OpenWeatherMap API**
  We used the `current weather` and `historical data` APIs from OpenWeatherMap to fetch real-time weather and historical weather data. This allowed us to display:
  - Current weather conditions (temperature, humidity, wind speed, etc.)
  - Hourly historical weather data for a specific city or geographic location.

  API requests are made by passing in the latitude and longitude of the location, along with an API key that authenticates our app's access to the data.

  Sample API call:

  ```
  https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API_KEY}
  ```

- **Leaflet API**
  This API was used for rendering interactive maps. Users can view the weather location

on a map, which is dynamically updated based on the city name or coordinates they input.

3. User Interaction

Users can interact with the application in the following ways:

1. **Enter a City Name:** Users type a city name into the search box to retrieve the weather data.
2. **Geolocation:** The app automatically detects the user's location and displays the weather data for that location, along with a map pinpointing their position.
3. **Weather Data:** After submitting a city name or using geolocation, the application fetches and displays the weather data, such as:
    - Current temperature
    - Humidity
    - Wind speed
    - A weather icon based on the current conditions
    - Time and date of the weather report

4. Challenges Faced During Development

Some of the challenges we encountered during the development process include:

- **API Limitations:** We faced rate limits while calling the OpenWeatherMap API. We had to ensure that our app doesn't exceed the free tier limits. We also had to manage the response format and handle cases where the API might return errors.
- **Geolocation Accuracy:** Although the Leaflet API provides a map, we had difficulty ensuring the accuracy of the geolocation feature in certain browsers. This required us to implement additional checks and fallback mechanisms.
- **UI Design:** We had to ensure the application's UI was both functional and aesthetically pleasing. This involved multiple iterations to get the weather data presentation just right.

5. Documentation

We have provided detailed documentation of the application, including:

- **API Integration:** The steps to integrate OpenWeatherMap API and Leaflet API have been clearly explained. The API calls and parameters are documented, and we have shown examples of how data is retrieved and displayed on the app.
- **User Interaction:** Instructions for users on how to search for a city or use geolocation to get weather data have been included.
- **Setup Instructions:** Steps for setting up the application, including obtaining an API key from OpenWeatherMap and configuring the app to use the APIs, are included.

The source code has been commented on in key sections to explain the functionality and logic behind each component.

6. Testing

We tested the integration of the APIs in several ways:

- **Unit Tests for Weather Data Fetching:** We wrote tests to ensure the weather data is correctly fetched from the OpenWeatherMap API and displayed on the app.
- **Geolocation Test:** We tested the geolocation feature to ensure it accurately detects the user's location and fetches the correct weather data.
- **Map Display Test:** We verified that the map displayed the correct location and updated when the user changed the city.

Test cases were conducted in different environments (e.g., different browsers and devices) to ensure the app worked smoothly.

7. Version Control

We used GitHub for version control throughout the project. Each member made meaningful contributions by pushing their code changes regularly. Our GitHub repository includes:

- **Complete Source Code:** All the files necessary for running the application are stored in the repository.
- **Commit History:** We documented each step of the development process with clear commit messages, reflecting the changes made at each stage.
- **Collaborative Work:** We ensured that all team members contributed to the project by reviewing each other's code and providing feedback.

The final project was submitted through a shared repository to ensure that all group members' contributions were included.

This report summarizes the development and key features of our Weather Application. By integrating OpenWeatherMap's weather and historical data APIs along with the Leaflet map API, we were able to create an interactive and useful weather app that provides users with accurate weather data and a visual map of their location.