Hackathon Set- 1: Weather Dashboard

Scenario

Imagine you are developing a simple weather application for travelers who need real-time weather updates before planning their trips. The app should allow users to enter a city name and receive the latest weather conditions, including temperature, humidity, and a short description (e.g., "Sunny" or "Rainy"). The app should be lightweight, fast, and accessible via a web browser.

Problem Statement

Build a Weather Dashboard that fetches real-time weather data from the OpenWeather API and displays it in a user-friendly web application. The application should allow users to:

- Search for a city's weather details.
- View temperature, humidity, and weather conditions.
- Display weather icons based on the conditions (e.g., sun for clear weather, clouds for cloudy weather).

Key Points to Consider

- Use OpenWeather API (or any free weather API) to fetch data.
- Implement a clean and responsive UI using HTML, CSS, and JavaScript (or React.js).
- Handle API errors gracefully (e.g., invalid city name should show an error message).
- Deploy the application on Render.com, Railway.app, or Netlify (free-tier hosting).
- Store the project on GitHub, with proper documentation in the README file.

Expected Output

- A deployed web application where users can search for any city's weather.
- Weather details such as temperature (°C/°F), humidity (%), and a short weather description.
- A visually appealing UI with weather-related icons.
- A GitHub repository with the source code and instructions for running the project.

Hackathon Set- 2: Handwritten Digit Recognition Web App

Scenario

Many AI-powered applications require digit recognition, such as automated form processing and handwritten check scanning. You are tasked with creating a simple web-based digit recognition system that allows users to draw a digit (0-9) on a canvas, and an AI model predicts the number.

Problem Statement

Develop a Handwritten Digit Recognition Web App where users can draw a number using their mouse or touchscreen, and a pre-trained deep learning model predicts the digit. The model should be trained on the MNIST dataset and deployed for real-time inference in a web browser.

Key Points to Consider

- Train a CNN-based model using TensorFlow/Keras on the MNIST dataset (or use a pre-trained model).
- Convert the trained model into TensorFlow.js format for use in a browser.
- Build a simple frontend using HTML, CSS, JavaScript (or React.js) with a drawing canvas.
- Integrate the model for real-time predictions when a user draws a digit.
- Deploy the application on GitHub Pages, Netlify, or Hugging Face Spaces (free hosting).

- Document the training process and model deployment in the GitHub repository.

Expected Output

- A deployed web app where users can draw a digit, and the model predicts the number.
- A simple and intuitive UI with a drawing canvas and a prediction label.
- A pre-trained model running in the browser (no server required).
- A GitHub repository with source code, model files, and setup instructions.