Weather Dashboard Project Report

👨‍💼 Intern Details

- Name: Venne Narayana Rao
- Intern ID: CT08DG1792
- Company: CodTech IT Solutions
- Domain: Java Programming
- Duration: 8 Weeks
- Mentor: Neela Santosh Kumar

📌 Project Title

- Weather Dashboard using REST API Client and Spring Boot

📖 Objective

- To create a weather dashboard web application that fetches and displays real-time weather data for a given city using the OpenWeatherMap API. This application uses Spring Boot for the backend and HTML + CSS for the frontend.

🔧 Technologies Used

- Java 17+
- Spring Boot 3.1.0
- Maven
- OpenWeatherMap API
- HTML, Tailwind CSS
- Git & GitHub

🧩 Modules

1. Backend API Integration (Spring Boot)
2. Frontend Web Dashboard
3. Static HTML Page Hosting in Spring Boot

🗂️ Project Structure

```
weather-dashboard/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/example/weather/
│       │       ├── WeatherApplication.java
│       │       ├── controller/
│       │       │   └── WeatherController.java
│       │       └── model/
│       │           └── WeatherResponse.java
│       └── resources/
│           ├── application.properties
│           └── static/
│               ├── dashboard.html
│               └── login.html
├── target/
├── pom.xml
└── README.md
```

📋 Steps to Run the Project

1. Start Spring Boot App

- mvn clean install
- mvn spring-boot:run

2.Open Frontend

- http://localhost:8085/dashboard.html

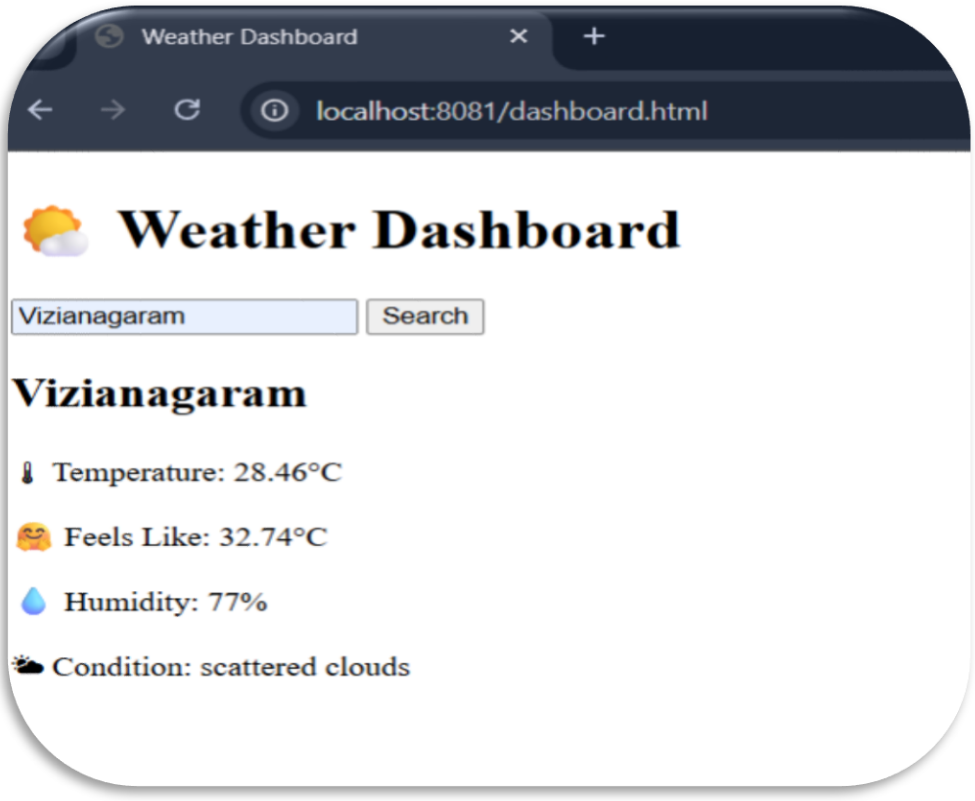
3.Enter City Name and View Weather

🔗 GitHub

Uploaded to: https://github.com/Narayanarao-v/weather-dashboard.git

Output Screenshot

- Shown working at: http://localhost:8085/dashboard.html
- Input: City Name
- Output: Weather, Temperature



➢ Project Setup

- The project is created using Spring Boot (Java) for the backend.
- It uses HTML + CSS for the frontend.
- The OpenWeatherMap API is used to fetch live weather data.

➢ Backend Initialization

- The backend application is started using a Spring Boot runner (WeatherApplication.java).
- When the app launches, it starts an **embedded Tomcat server** on a specific port (e.g., 8085).
- It loads configuration from the application.properties file — including the server port and weather API key.

➢ API Communication

- The backend includes a **Weather Controller** which handles HTTP requests from the frontend.
- When a user submits a city name in the frontend form:
    - A **GET request** is sent to the backend.
    - The backend processes the request and **sends a request to the OpenWeatherMap API**.
    - The response (weather details like temperature, humidity, condition) is **extracted** and structured in a simple format.

➢ Frontend Interaction

- The frontend consists of:
    - login.html: A basic login page (optional).
    - dashboard.html: The main weather page.
- The user enters a **city name** in the input box on the dashboard.
- When the user clicks the **submit/search** button:
    - A JavaScript function sends the request to the backend.
    - The frontend receives the weather data in response and **updates the dashboard UI** dynamically.

Output Display

- The dashboard updates to show:
    - Current temperature
    - Weather condition (like sunny, rainy)
    - Humidity and pressure (if included)
- It may display this data in cards, boxes, or stylized sections using CSS.

Project Success Check

You know the project is working when:

- Visiting http://localhost:8085/dashboard.html shows the dashboard.
- Submitting a city like “Hyderabad” returns live weather info.
- The terminal shows Spring Boot logs confirming the backend handled the request.

📌 Conclusion

This project helped in understanding:

- Spring Boot application setup
- Calling third-party APIs (OpenWeatherMap)
- Creating and hosting static web pages with backend logic
- Using Maven and GitHub for building and deployment