**Infoway Technologies Pvt. Ltd, Pune CDAC PG-DAC**

Project Name: **Weather Prediction Web Application**

Project Members:

1.Akanksha Pradeep Chavan 250243120006

2.Mayuri Rajendra Patil 250243120031

3.Mrunal Rajendra Desai 250243120033

**Index**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Title | Page No. |
| 1 | Introduction |  |
| 2 | Problem Definition & Scope |  |
| 2.1 | Problem Definition |  |
| 2.2 | Goals & Objectives |  |
| 2.3 | Major Constraints & Outcomes |  |
| 3 | Software Requirement Specification |  |
| 3.1 | Purposed System |  |
| 3.2 | Scope |  |
| 4 | System Modules |  |
| 5 | Performance Requirements |  |
| 5.1 | H/W Requirements & S/W Requirements |  |
| 6 | UML Diagram |  |
| 6.1 | DFD |  |
| 6.2 | ERD |  |
| 6.3 | Use case diagram |  |
| 6.4 | Class Diagram |  |
| 6.5 | Sequence diagram |  |
| 6.6 | Activity Diagram |  |
| 6.7 | Deployment diagram |  |
| 6.8 | System Architecture |  |
| 7 | Test Cases | 6 |
| 8 | Screenshots like Landing page, Login, Orders, payment, different type Reports, etc |  |
| 9 | References |  |

Introduction

Weather prediction, or weather forecasting, is the process of using science and technology to determine the future state of the atmosphere for a particular location and time. Although people have been trying to forecast weather for centuries using signs from nature, modern weather forecasting officially began in the 19th century. In the early days, forecasts were based on observations like barometric pressure, cloud patterns, and previous weather behavior. Today, weather forecasting has become much more advanced, relying on powerful computer models that analyze a wide range of atmospheric data.

Modern weather forecasting combines real-time data collection with meteorological science to predict temperature, humidity, wind speed, rainfall, and cloud cover. While computers play a big role in processing and analyzing this data, human forecasters still have an important part in interpreting results and choosing the most accurate models.

Weather forecasting has significant practical value. For example, in 2009, the United States invested $5.1 billion in weather forecasting, with estimated returns nearly six times that amount. Accurate forecasts can help save lives, protect property, and support industries like agriculture, aviation, and shipping

This project presents a web-based weather forecasting application that provides real-time atmospheric information for cities across the globe. The system uses an API to gather and process data, offering reliable forecasts that users can access anytime, anywhere. The application focuses on key weather elements such as temperature, humidity, wind speed, rainfall, and cloud coverage. The collected data can also be stored and analyzed for future reference.

Weather is one of the most influential natural phenomena affecting our daily lives. Since it changes constantly due to climate, seasonal shifts, and other factors, having access to accurate forecasts is incredibly useful. The main aim of this project is to make global weather data easily accessible and understandable through a user-friendly web application.

Working on this project has been a valuable experience, helping me enhance both my technical skills and understanding of real-world applications. I’m grateful for the opportunity to develop this system, which reflects the growing importance of combining science and technology in everyday life.

**2.Problem Definition & Scope**

**2.1 Problem definition**

Developing this weather forecasting project came with its own set of challenges. One of the major difficulties was **collecting accurate historical weather data**. Reliable past data is essential for training and testing forecasting models, but sourcing it in a consistent format from trustworthy providers required time and careful validation.

Another significant challenge was **establishing a stable real-time API connection** to fetch current weather data. While collecting data from populated land areas is relatively straightforward, gathering accurate real-time weather information over oceans or in the atmosphere is much more difficult. These areas are hard to monitor continuously due to their inaccessibility—people don't live in the middle of the ocean or high up in the air, and placing instruments there is expensive and complex.

Additionally, **geographical features pose another layer of difficulty**. Locations such as coastal regions or mountainous areas experience rapid changes in weather over very short distances. Capturing these localized changes accurately requires a dense network of observation points, which is often not practical or feasible due to cost, terrain, and accessibility.

Despite these hurdles, integrating real-time weather APIs and building a system that can process, display, and store data accurately has been a valuable learning experience. These challenges highlighted the complexities behind what might seem like a simple weather update on a phone app and deepened my appreciation for the science and technology that goes into making accurate weather forecasts possible.

**2.2 Goals and Objectives**

**Goals**

The main goal of this project is to develop a web-based weather forecasting application that provides accurate, real-time weather updates for any city across the world. The system aims to make weather information easily accessible to users by utilizing an external API to fetch and display key atmospheric data such as temperature, humidity, wind speed, rainfall, and cloud coverage.

This project also aims to:

* Help users make informed decisions based on current and forecasted weather conditions.
* Offer a simple and user-friendly interface for checking weather updates from anywhere at any time.
* Store and manage weather data for future reference or analysis.
* Improve understanding of how real-time data collection, APIs, and forecasting models work together to generate weather predictions.

**Objective**

The primary objective of this project is to design and develop a **web-based weather report application** that provides accurate and real-time weather forecasts for any location around the world. The application uses browser-based geolocation to automatically detect the user’s location and display key weather details such as **temperature, wind direction, rainfall, humidity**, and more. Users can also manually select different locations to view weather data for other areas.

This system integrates with weather APIs to pull data directly from satellite and radar sources, ensuring reliable and up-to-date information. Advanced features such as **animated weather widgets**, **email notifications**, and **critical weather alerts** enhance the user experience. Even when users are offline or the server is temporarily down, they can still receive weather updates directly to their email through the app’s feed and alert system.

The application is designed to be highly **user-friendly**, with a simple interface that makes it accessible to users of all ages, even children. The goal is to combine **accurate forecasting** with **ease of use**, providing users with a dependable tool to monitor the weather conditions of any geographical location at any time.

A key technical objective of the project is to apply and enhance **Full-Stack Development skills**, including working with **front-end, back-end, APIs, and real-time data handling**. This project provides hands-on experience in integrating client-server communication, Java servlet coding, and responsive web design.

Weather forecasting plays a vital role in everyday life and affects various sectors. This application demonstrates the importance of forecasting through practical use cases such as:

* Helping individuals dress appropriately for current weather.
* Assisting in energy planning and power consumption.
* Supporting outdoor event planning by predicting rain, wind, or storms.
* Preparing for severe weather conditions such as snow, lightning, hail, or hurricanes.
* Assisting people with health issues like asthma or allergies.
* Enabling safe transportation planning by predicting fog, icy roads, or storm conditions.
* Helping farmers and gardeners manage irrigation and protect crops from weather damage.

In summary, this project not only serves as a useful real-world application but also enhances the developer's understanding of web technologies, data integration, and the impact of weather forecasting on society.

**2.3 Major Constraints** **and** **Expected Outcomes**

**Major Constraints**

1. **Dependence on Third-Party APIs**  
   The system relies heavily on external weather APIs for real-time data. Any limitations in the API—such as data accuracy, request limits, latency, or unexpected downtime—can affect the application's performance and reliability.
2. **Internet Connectivity**  
   As a web-based application, continuous internet access is required. Users without a stable internet connection may face difficulties in accessing updated forecasts.
3. **Limited Coverage in Remote Areas**  
   Weather data may not be as precise in remote or under-monitored regions such as deep oceans, mountain ranges, or less-populated rural areas due to sparse data collection points.
4. **Accuracy of Forecasts**  
   Although the system uses advanced models and real-time data, weather is inherently unpredictable. Sudden changes in climate conditions may not always be captured in time, which could lead to slight inaccuracies.
5. **Cross-Browser and Device Compatibility**  
   Ensuring consistent performance across different web browsers, operating systems, and screen sizes may require additional optimization and testing efforts.

**Outcomes**

1. **User-Friendly Weather Forecasting Platform**  
   A clean, responsive, and easy-to-use web application that provides users with reliable and timely weather information for any global location.
2. **Geolocation-Based Forecasts**  
   Automatic location detection through browser settings for quick access to local weather, with the ability to manually search for and monitor other locations.
3. **Real-Time Data Integration**  
   Live weather data including temperature, humidity, wind speed, rain prediction, and cloud coverage displayed using real-time API connections.
4. **Educational and Technical Growth**  
   Improved understanding of full-stack development principles, including API integration, backend processing, client-server interaction, and UI/UX design.
5. **Scalability for Future Enhancements**  
   A flexible architecture that can be expanded in the future with features like multi-day forecasts, historical data comparison, voice assistance, or mobile app support.

**3. Software Requirement Specification**

### ****3.1 Proposed System****

The proposed weather forecasting system is a **web-based application** designed to provide real-time, accurate weather information for users across the globe. The system aims to improve the user experience by offering timely updates and practical features that support everyday planning and decision-making.

#### **Key Features of the Proposed System:**

* **Real-Time Weather Updates**  
  The system fetches weather data in real-time, ensuring that users always have access to the most current weather conditions.
* **Temperature Monitoring**  
  Live temperature updates are displayed and refreshed periodically to reflect actual atmospheric changes.
* **Hourly Forecast Changes**  
  The application is designed to automatically update and reflect changes in the weather every hour, based on dynamic API data.
* **7-Day Weather Forecast**  
  Users can view weather predictions for the upcoming 7 days, helping them plan activities, travel, or events in advance.
* **Accurate Weather Information**  
  The system provides reliable data on temperature, humidity, rainfall, wind speed, and more—sourced from trusted weather APIs.
* **Search Weather by Location**  
  Users can search and view the weather conditions of any city or location worldwide, at any time.
* **Location Flexibility**  
  Whether you're at home or abroad, the system allows users to explore weather data for any place of interest with ease.
* **Travel Assistance**  
  By knowing the expected weather, users can make smarter travel decisions, avoiding severe weather or choosing optimal travel days.
* **Holiday Planning Support**  
  The app helps users plan holidays and outdoor activities by providing future weather forecasts that aid in choosing suitable dates and destinations.

**3.2 Scope**

There are various uses of weather forecasting in day-to-day life, it can be as simple as deciding whether to take an umbrella with you on your work or to deciding your outfit. Following are some of the places where weather forecasting plays a major role:

• Seasons and nature play a major role in agriculture and farming. When it comes to the farming of various fruits, vegetables, and pulses, temperature is extremely important. Farmers didn't have a better understanding of weather forecasts before, so they had to rely on estimates to do their jobs. They do, however, sometimes suffer losses as a result of inaccurate weather forecasts. Farmers will now get all of their forecasts on their smartphones, thanks to advances in technology and the use of unique weather forecasting mechanisms. Of course, education in this area is critical, but the majority of the farmer community at this point understands the fundamentals, making it simple for them to use the features. It aids food grain transportation and storage. It aids in the handling of cultural operations such as harrowing, hoeing, etc. It aids in the implementation of livestock protection initiatives.

• Weather Forecasting is crucial since it helps to determine future climate changes. With the use of latitude, we can determine the probability of snow and hail reaching the surface.

We are able to identify the thermal energy from the sun that is exposed to a region. Climatology is the scientific study of climates, which in simple words mean weather conditions over a period. A bunch of studies within atmospheric sciences also takes the help of the variables and averages of short-term and long-term weather conditions accumulated. Climatology is different from meteorology and can be divided into further areas of study. Different approaches to this segment can be taken. Currently, our primary goal is to motivate and develop efficient and effective 8 measures of Environmental activities.

**4. System Module**

The weather forecasting system is divided into several key modules, each responsible for handling specific tasks within the application. These modules work together to provide a seamless and efficient user experience.

#### 1. **User Interface Module**

* Provides a clean and responsive interface for users to interact with the application.
* Displays real-time weather data including temperature, humidity, wind speed, rain chances, and cloud cover.
* Allows users to search weather conditions for different cities or locations.
* Supports dark/light themes or widgets (optional enhancement).

#### 2. **Location Detection Module**

* Automatically detects the user’s current location using browser-based geolocation.
* Sends location data to the backend or weather API to fetch relevant weather information.
* Offers manual location selection if geolocation is disabled or unavailable.

#### 3. **Weather Data Fetching Module**

* Connects with external weather APIs (e.g., OpenWeatherMap, WeatherAPI) to retrieve live weather data.
* Handles API requests and responses securely and efficiently.
* Updates data at regular intervals (e.g., every hour) to ensure freshness.

#### 4. **Historical Data Module**

* Fetches and displays weather trends from the past 7 days.
* Helps users compare current conditions with recent history.
* Useful for analyzing weather behavior and patterns over time.

#### 5. **Forecast Module**

* Provides weather predictions for the upcoming 7 days.
* Shows daily temperature highs and lows, rain chances, wind conditions, and more.
* Updates forecast data based on the latest API responses.

#### 6. **Notification/Alert Module**

* Sends alerts for severe weather conditions (e.g., storms, heavy rain, heatwaves).
* Optional: integrates email notifications or push alerts to keep users informed even when offline

### ****5.Performance Requirements****

The performance of the weather forecasting application is critical to ensuring a smooth, responsive, and accurate user experience. The following performance requirements define the expected system behavior under normal and peak conditions.

#### **1. Real-Time Data Retrieval**

* The system must fetch and display updated weather data within **2–3 seconds** of the user's request.
* The API call should be processed efficiently to avoid noticeable delay in data loading.

#### **2. Fast Page Load Time**

* The application should load the main dashboard (including geolocation-based weather info) in **less than 5 seconds** under normal internet conditions.
* Lazy loading or caching strategies should be used to minimize initial load times.

#### **3. API Response Time**

* API responses (from weather providers) should be processed and rendered within **1 second** of receiving the data.
* Failover or fallback mechanisms should be in place in case of delayed or failed API responses.

#### **4. Scalability**

* The system should be able to handle multiple simultaneous users without performance degradation.
* Backend should be optimized to support high traffic during peak weather updates (e.g., storms, heatwaves).

#### **5. Accuracy of Data**

* The system must ensure **at least 95% data accuracy** by using reliable third-party APIs and ensuring correct parsing of JSON responses.
* Hourly and daily forecast data must reflect current atmospheric conditions as closely as possible.

#### **6. Resource Utilization**

* The application should have optimized memory and CPU usage, especially for animations or background data fetching.
* Frontend performance should remain stable even on devices with limited resources.

#### **7. Responsiveness**

* The application must remain fully functional and responsive across different screen sizes (desktop, tablet, mobile).
* UI actions like location search, forecast navigation, and widget updates should execute within **300–500 milliseconds**.

#### **8. Uptime and Availability**

* The system should maintain **99.5% uptime**, ensuring continuous availability.
* Critical weather updates (e.g., storm alerts) should be prioritized and delivered without delay.

**5.1 H/W Requirements & S/W Requirements**

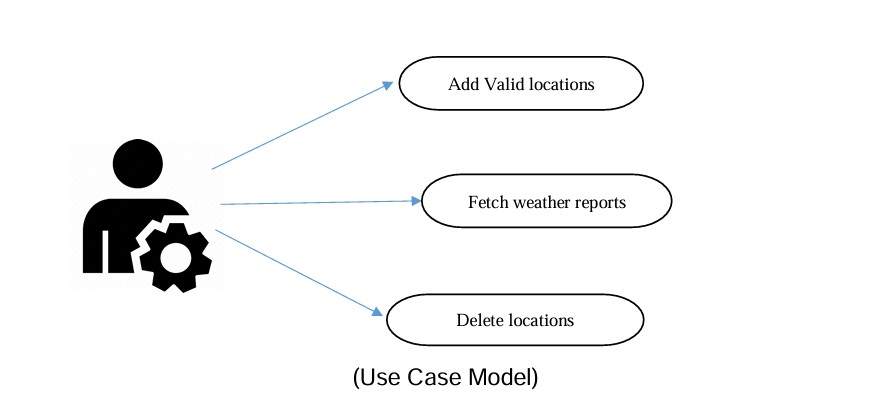
* **H/W Requirements**

| **Component** | **Minimum Requirement** | **Recommended Requirement** |
| --- | --- | --- |
| **Processor** | Intel Core i3 or equivalent | Intel Core i5 or higher |
| **RAM** | 4 GB | 8 GB or more |
| **Storage** | 10 GB free space | SSD with 20 GB free space |
| **Display** | 1366 × 768 resolution | Full HD (1920 × 1080) |
| **Internet** | Basic broadband (2 Mbps) | High-speed connection (10+ Mbps) |
| **Input Devices** | Keyboard, Mouse | Keyboard, Mouse |
| **Others** | Webcam (for testing UI responsiveness) | Optional |

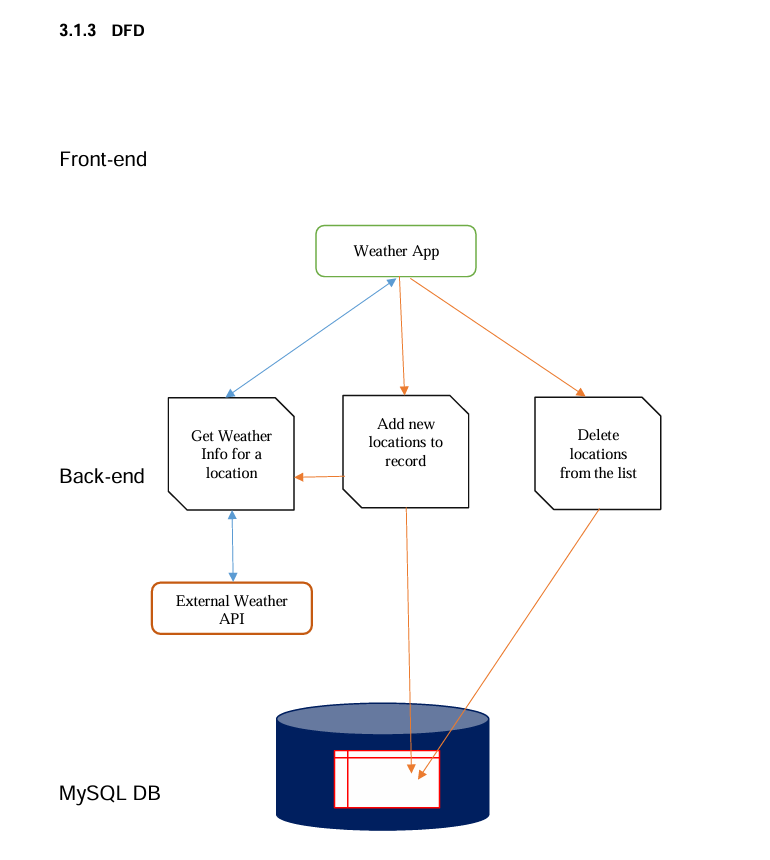
* **S/W Requirements**

| **Category** | **Software/Tool** | **Version / Notes** |
| --- | --- | --- |
| **Frontend** | HTML, CSS, JavaScript, React | React 18+, Vite |
| **Backend (Weather)** | Java, Spring Boot | Java 17+, Spring Boot 3.x |
| **Backend (Auth)** | .NET Core API | .NET 6 or later |
| **Database** | MySQL | MySQL 8.0+ |
| **API** | OpenWeatherMap API | Free/Pro API key |
| **Authentication** | JWT (JSON Web Tokens) | Using .NET Core |
| **IDE / Editor** | VS Code, IntelliJ, Visual Studio | Any latest stable version |
| **Version Control** | Git | Git CLI or GitHub Desktop |
| **Operating System** | Windows 10 / macOS / Linux | Any modern OS |
| **Browser** | Chrome / Firefox | Latest version |
| **API Testing Tool** | Postman | Optional |
| **Node.js & npm** | Node.js | Node 18+ with npm |
| **Java SDK** | Oracle OpenJDK | Version 17+ |
| **Build Tools** | Maven (for Java) / dotnet CLI | Maven 3.6+, .NET CLI |

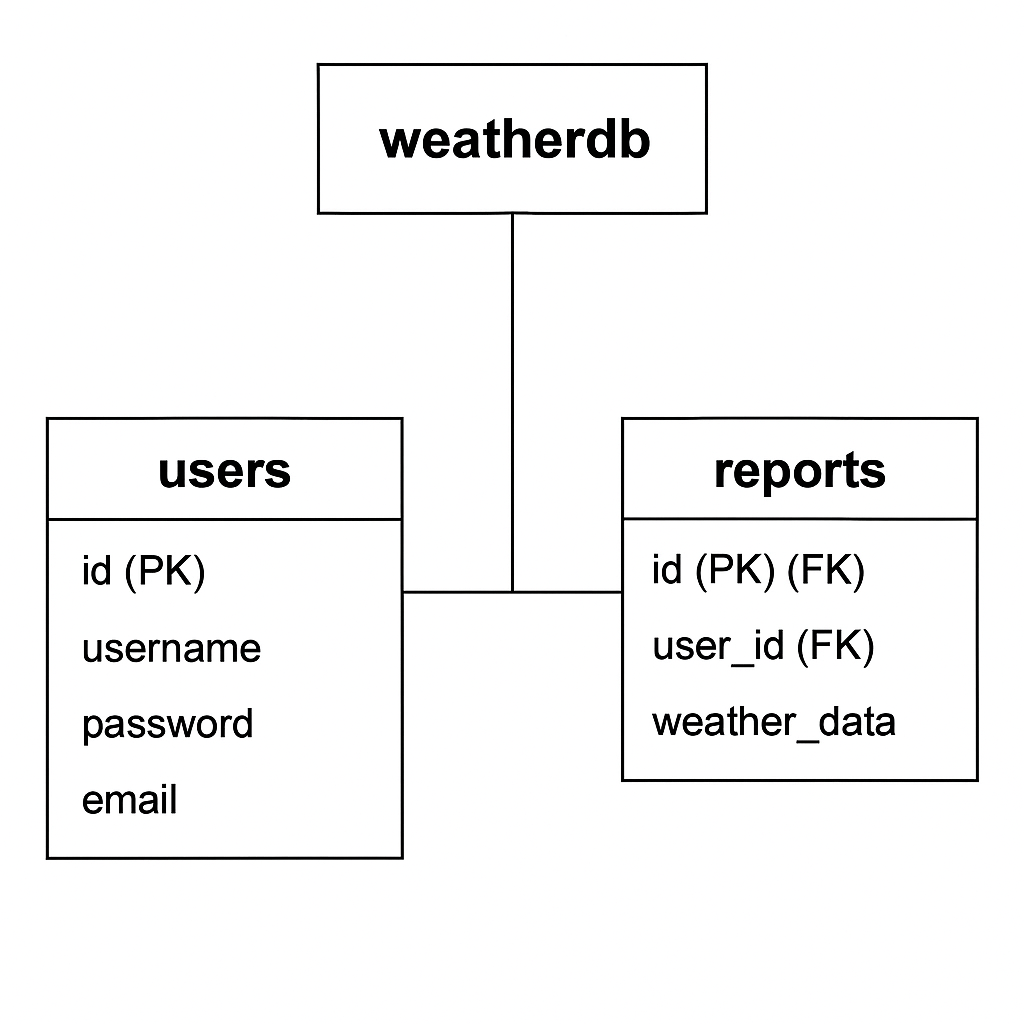
**6.UML Diagram**



**6.1 DFD**

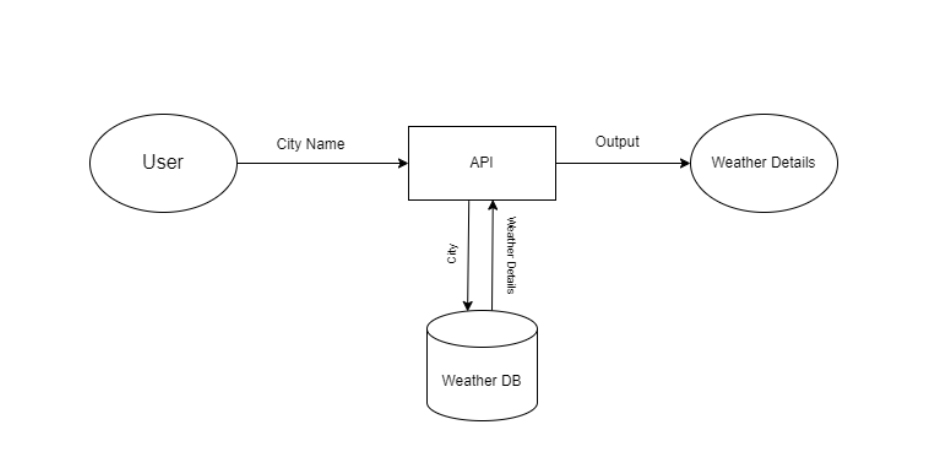


**6.2 ERD**

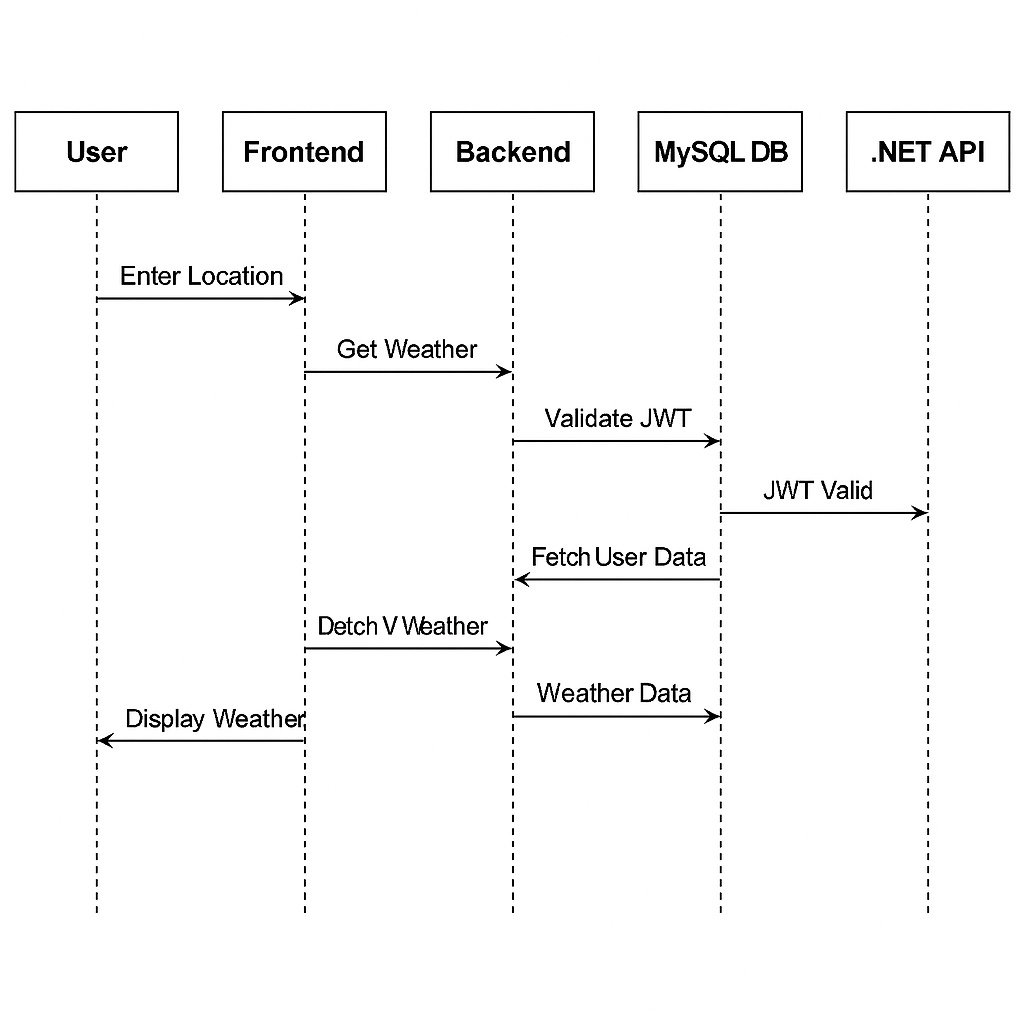
****

**6.2 Use Case**

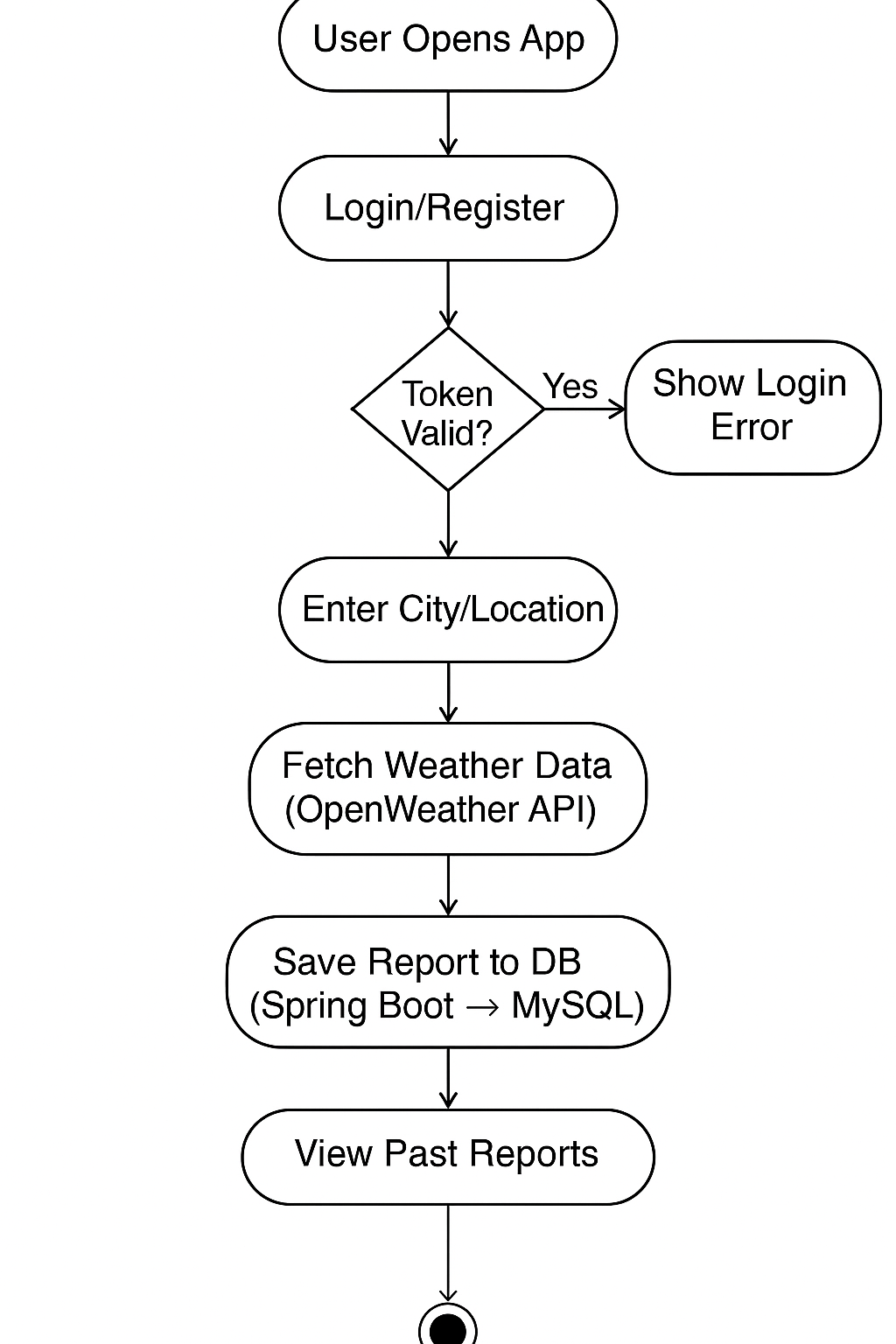


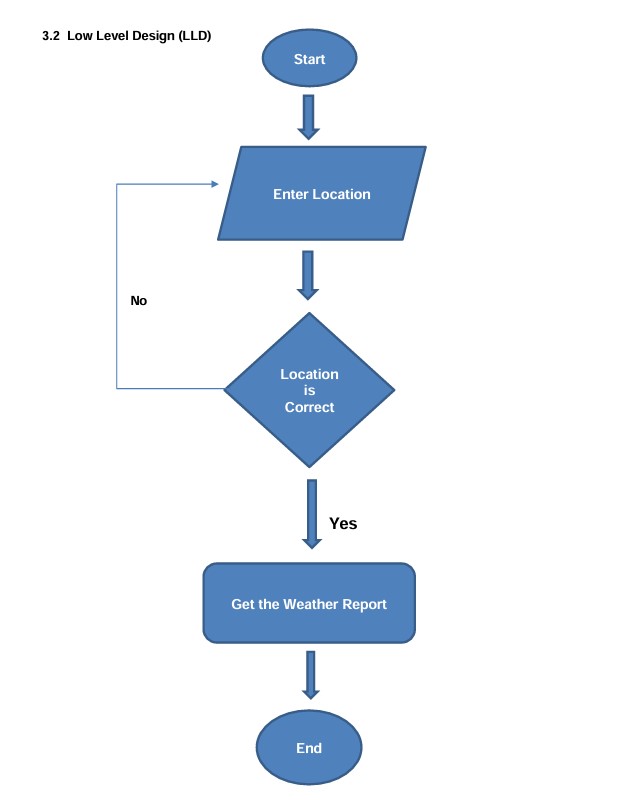


**6.5 Sequence Diagram**

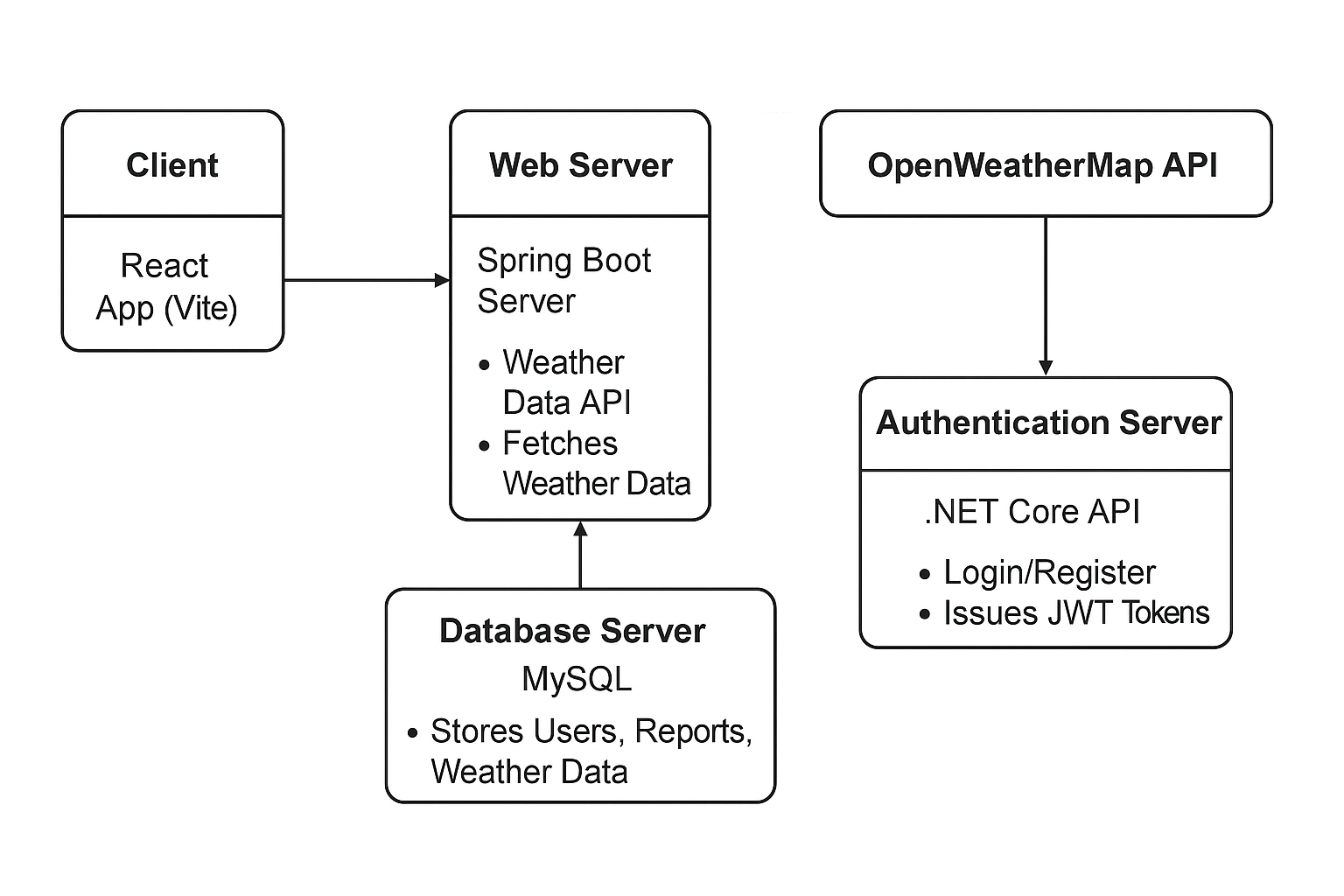
****

**6.6 Acitivity Diagram**

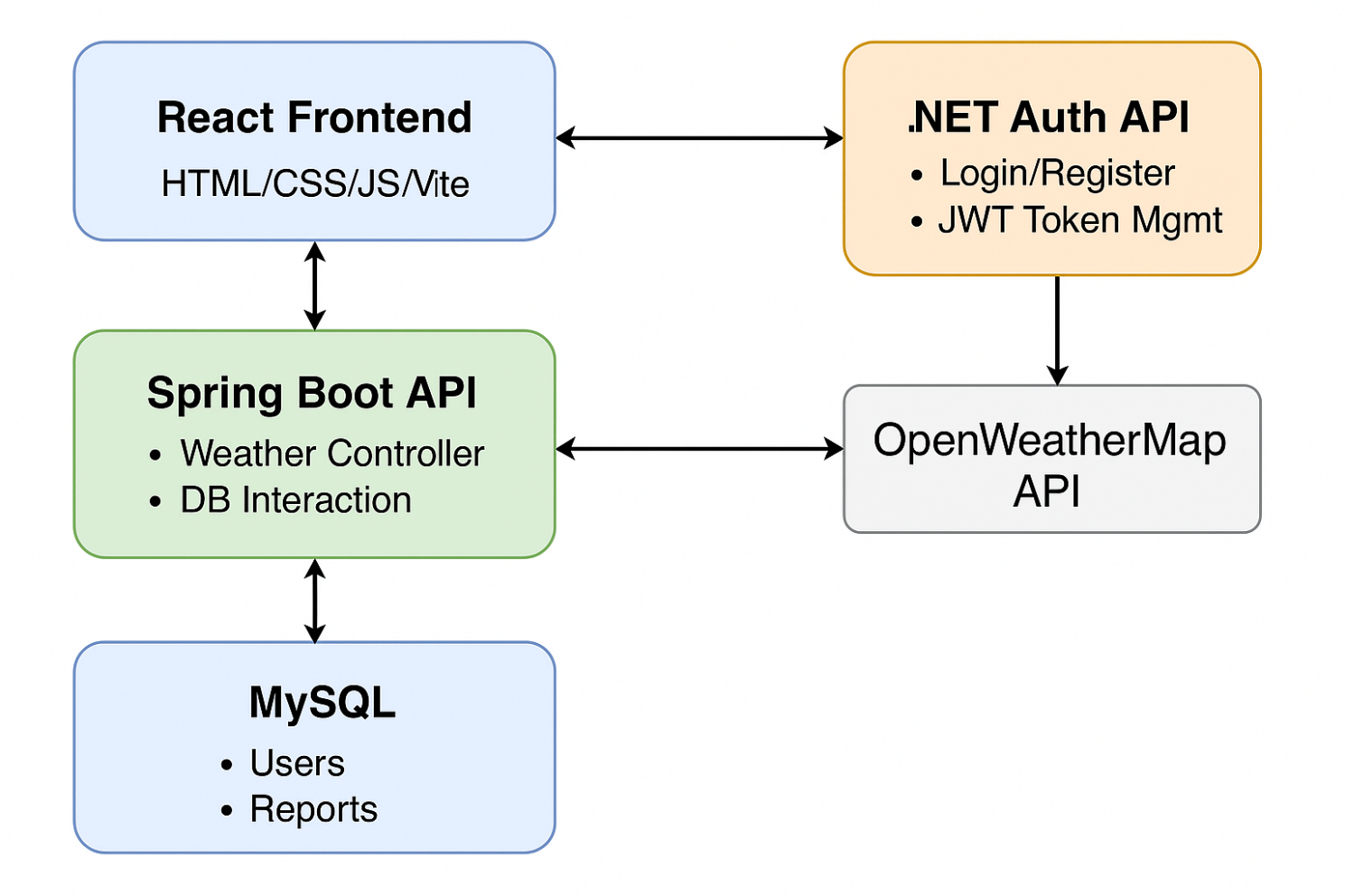
****



**6.6 Deployment Diagram**

****

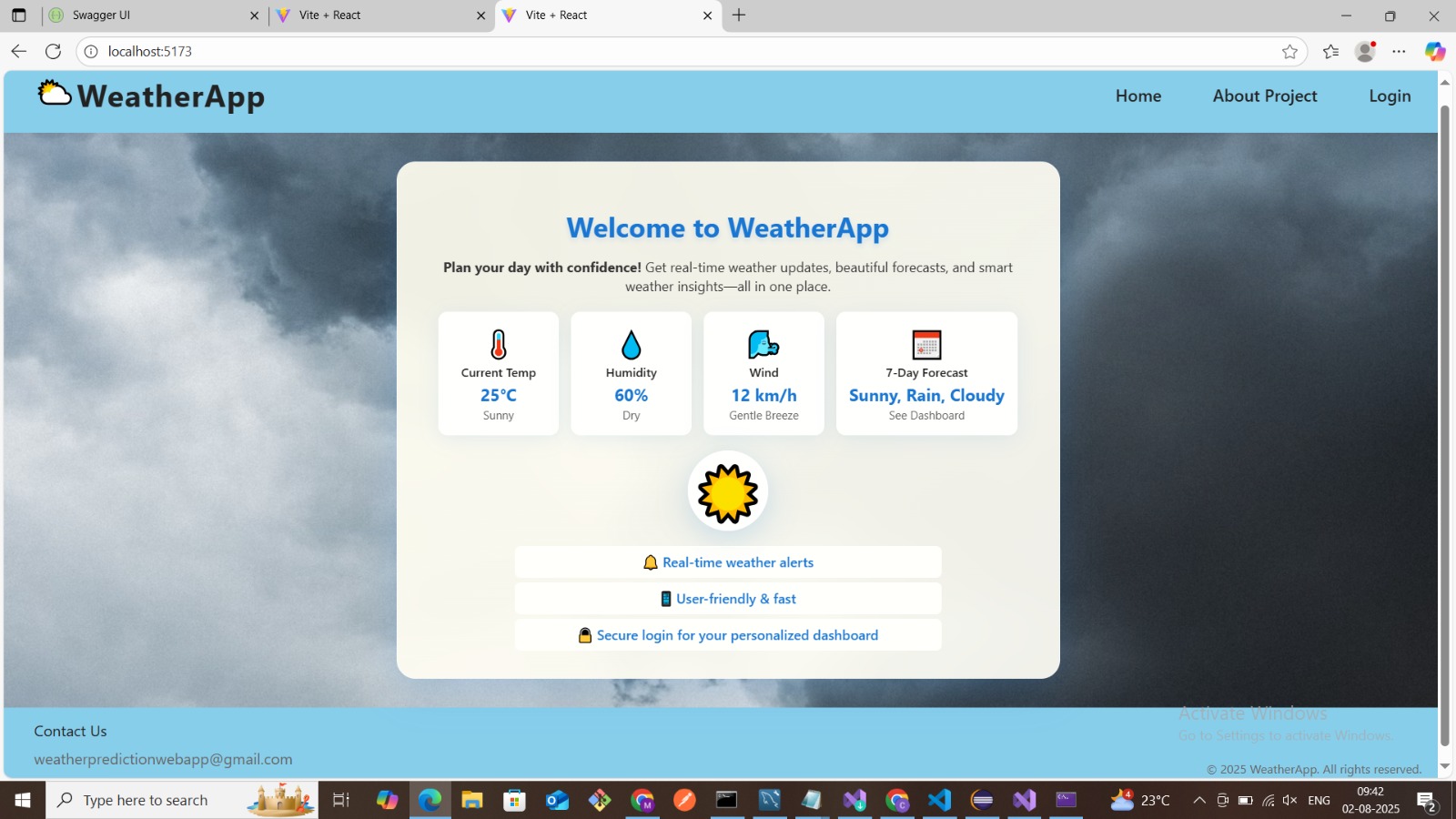
**6.8 System Architecture**

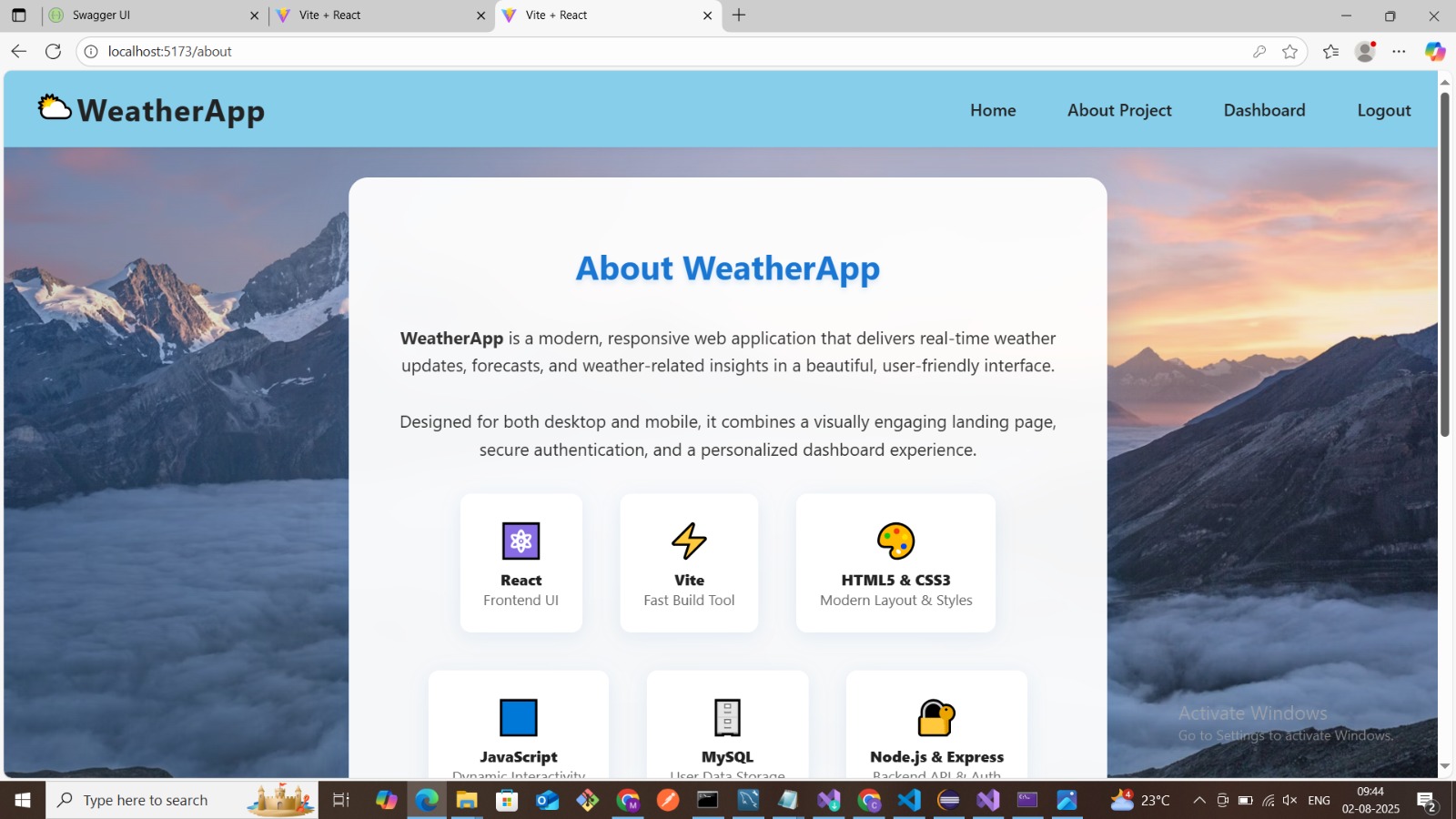
****

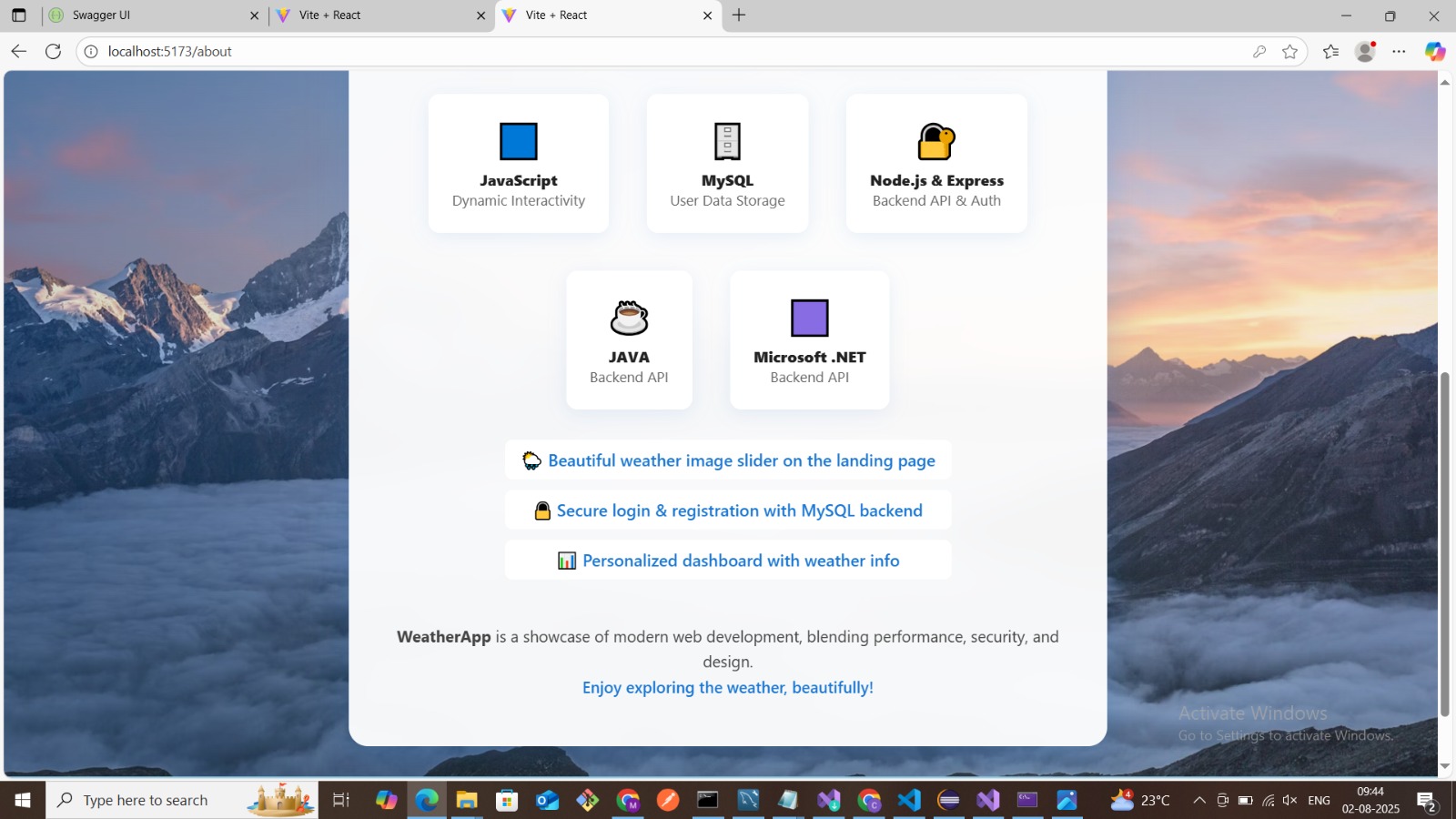
**7.Test Cases**

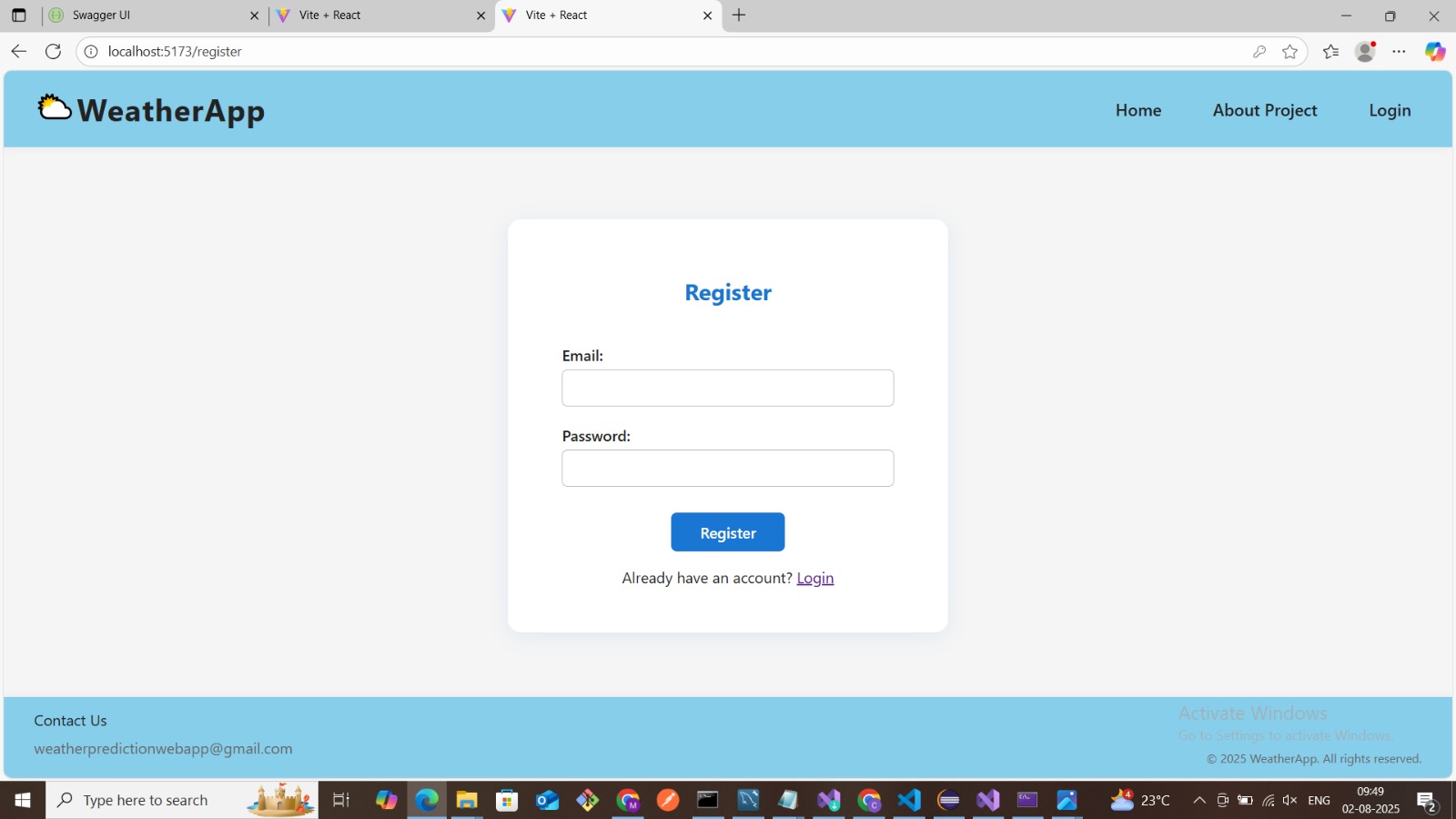
| **Test Case ID** | **Test Case Description** | **Preconditions** | **Test Steps** | **Expected Result** | **Priority** |
| --- | --- | --- | --- | --- | --- |
| TC001 | Verify Login with valid credentials | User is registered | 1. Navigate to Login page2. Enter valid email & password3. Click Login | User should be redirected to the dashboard/home page | High |
| TC002 | Verify Login with invalid password | User is registered | Enter valid email and wrong password, click Login | Error message should appear: “Invalid credentials” | High |
| TC003 | Verify Login with empty fields | None | Leave email and/or password blank, click Login | Error message should appear for required fields | High |
| TC004 | Verify Registration with valid inputs | None | Navigate to Register page, fill valid info, click Register | User account should be created, redirect to login or dashboard | High |
| TC005 | Verify Registration with existing email | Email already registered | Register with the same email as existing user | Error: “Email already in use” | Medium |
| TC006 | Check Register button disabled for empty/invalid form | None | Leave required fields empty or invalid | “Register” button should remain disabled | Medium |
| TC007 | Verify navigation to Login from Register page | On Register page | Click on “Already have an account? Login” link | User should be navigated to Login page | Low |
| TC008 | Verify navigation to Register from Login page | On Login page | Click on “Create new account” link | User should be navigated to Register page | Low |
| TC009 | Verify Home button redirects correctly | Logged-in user | Click on Home or logo | User should be navigated to the homepage | Medium |
| TC010 | Check Logout functionality | Logged-in user | Click Logout | User should be logged out and redirected to login/home page | High |
| TC011 | Check if unauthorized user can access dashboard | Not logged in | Directly enter dashboard URL in browser | Should be redirected to login page | High |
| TC012 | Check password strength validation | On Register page | Enter weak password | Warning should appear (e.g., use 8+ chars, 1 symbol) | Medium |

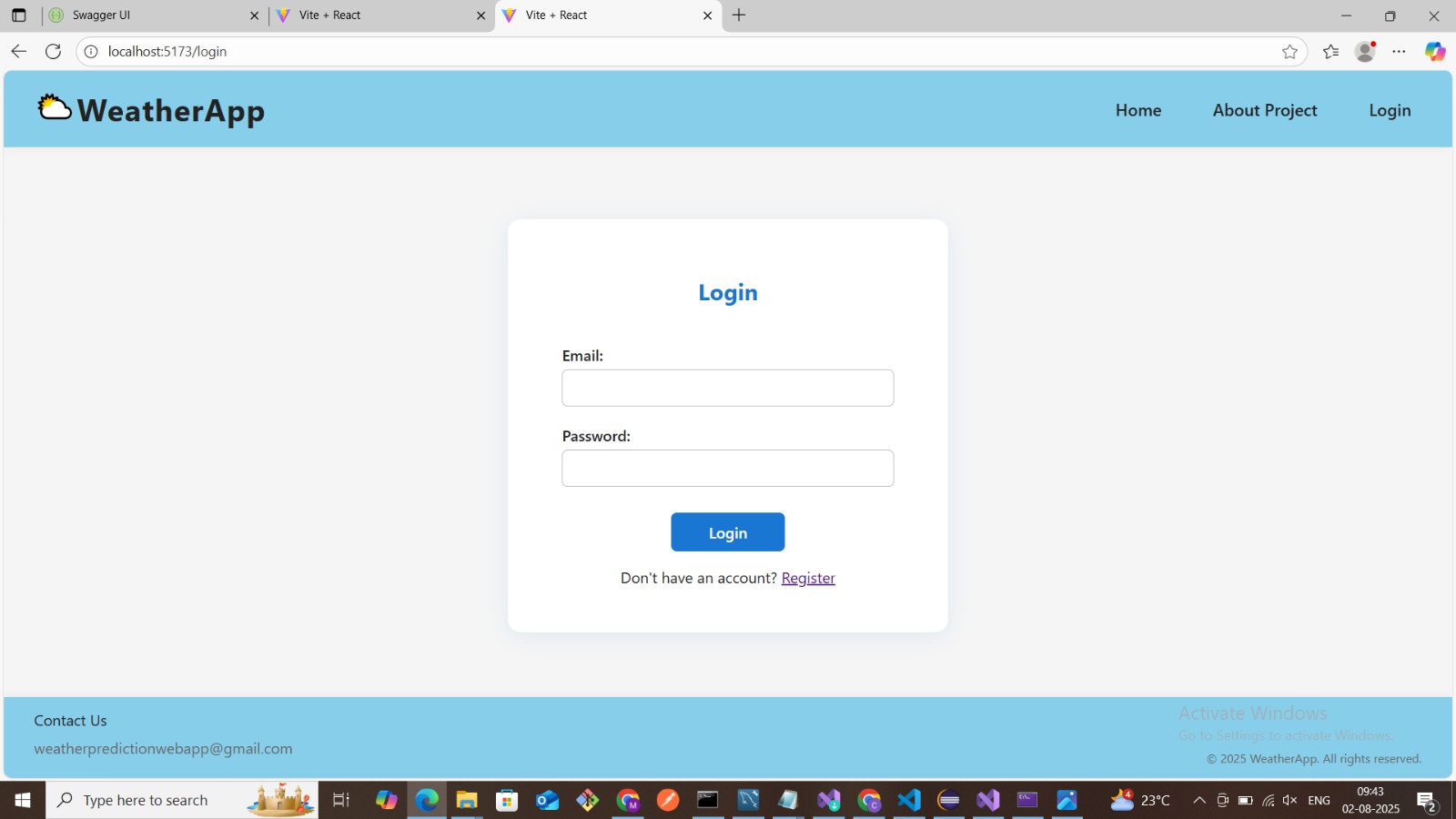
**8.Scrennshots**

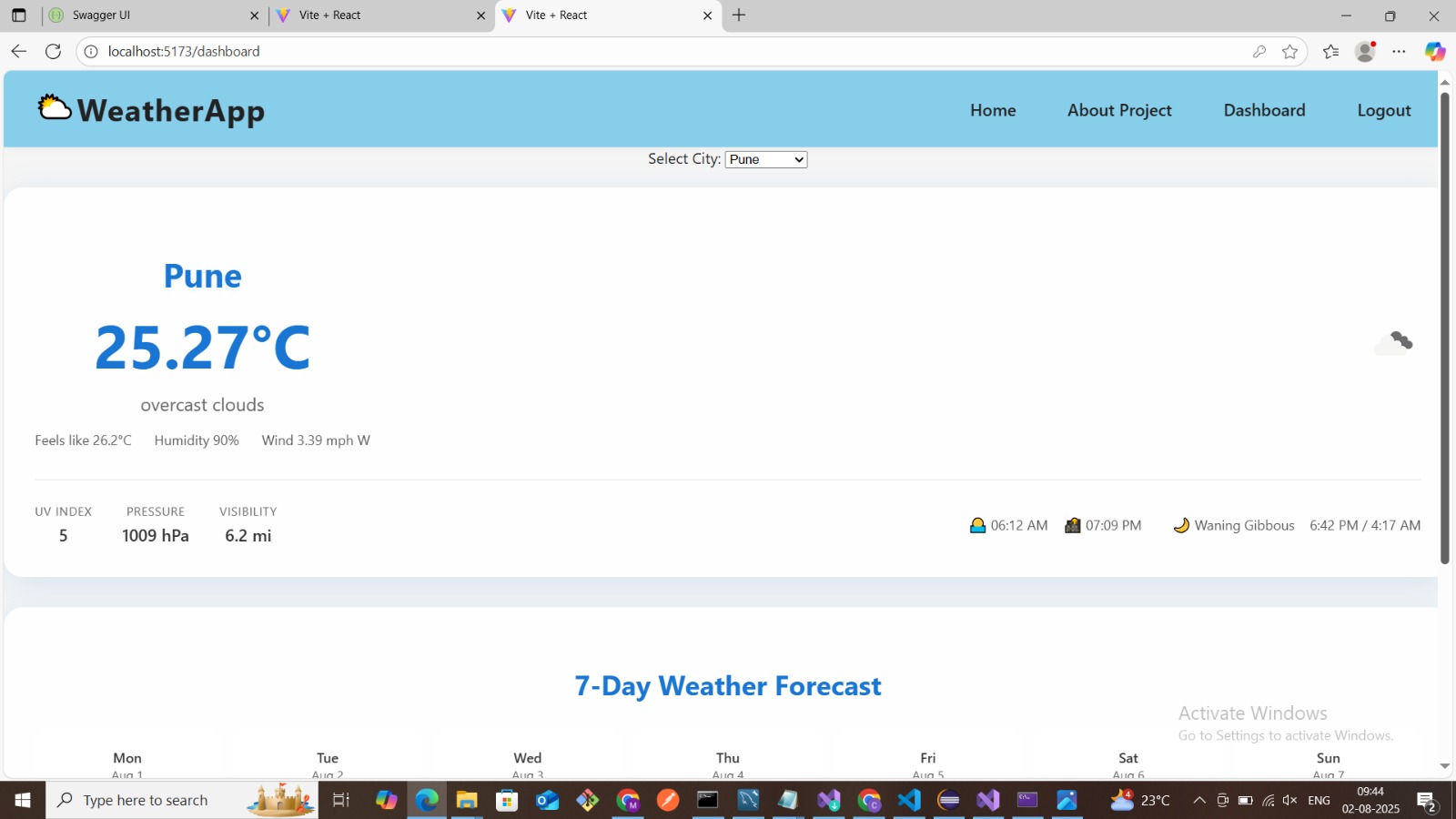
****

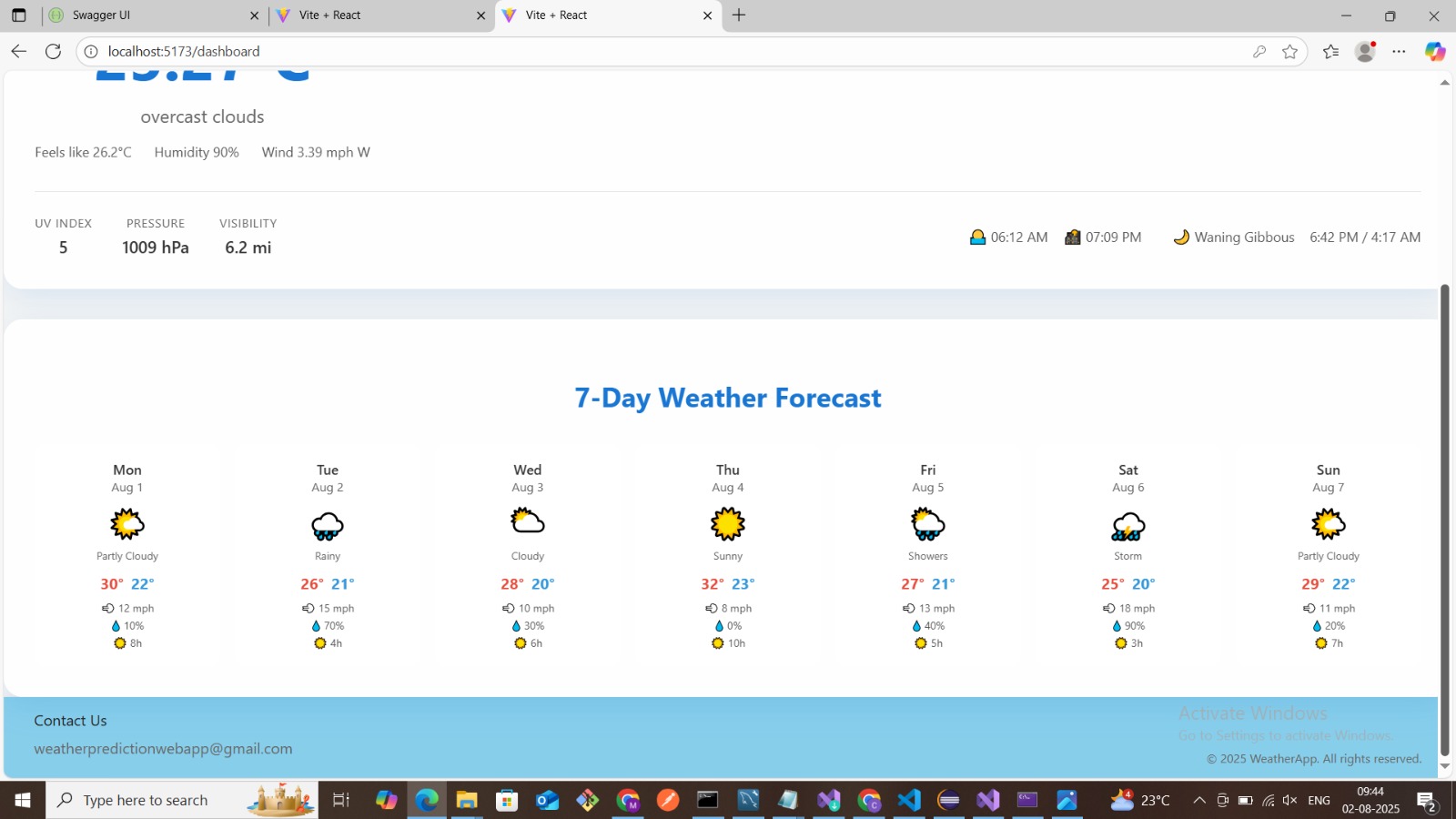
****

****

****

****

****

****

References:

<http://www.w3Schools.com>

<http://www.tutorialspoint.com>

<http://www.youtube.com>

<http://reactjs.org>

<http://nodejs.org>

<http://openearth.com>

<http://developer.mozilla.org>

http://google.com