WEATHER FORCAST

A PROJECT REPORT

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by

Rwitabrata Mandal : (24BCS80041)
Saksham Chandel : (23BCS13513)
Vaibhav Singh. : (23BCS11521)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Chandigarh University, Gharuan

November 2025

BONAFIDE CERTIFICATE

Certificate that this project report “WEATHER FORECAST” is the Bonafide work of “Rwitabrata ,Saksham,Vaibhav “ who carried outr the project workik under my/our supervision

HEAD OF THE DEPERTMENT
Er. Gagandeep Singh.

SUPERVISOR
Kamal Kumar

Submitted for thr project viva – voce examination held on 14 Nov 2025

INTERNAL EXAMINER

EXTERNAL EXAMINER

WeatherApp USING JAVA SERVLET JSP

Description

WeatherApp is a simple Java web application developed using Servlets, JSP, HTML, CSS, and JavaScript. It integrates with the OpenWeatherMap API to fetch weather data for a given city and display it to the user.

Features

Fetch weather data based on the user's input city name. Display current weather conditions including temperature, humidity, wind speed, visibility, and cloud cover, etc.

Technologies Used

- Java Servlets
- JavaServer Pages (JSP)
- HTML
- CSS
- JavaScript
- Gson library for JSON parsing
- OpenWeatherMap API

Setup Instructions

1. Download and install Eclipse IDE (or IntelliJ IDEA).
2. Download and install Apache Tomcat 10.1.1.
3. Open Eclipse IDE and configure it with Apache Tomcat:
    - Go to `Window` -> `Preferences`.
    - Navigate to `Server` -> `Runtime Environments`.
    - Click `Add` and select `Apache Tomcat v10.1.1`.
    - Provide the Tomcat installation directory and finish the setup.
4. Clone the repository to your local machine using `git clone <repository_url>`.
5. Import the project into Eclipse IDE:
    - Go to `File` -> `Import`.
    - Select `Existing Projects into Workspace`.
    - Choose the cloned project directory and import it into Eclipse.
6. Ensure that the Gson library is included in the `src/webapp/WEB-INF/lib`directory of your project. If not, add it manually to the build path.
7. Obtain an API key from OpenWeatherMap and replace the placeholder `myApiKey` in `MyServlet.java` with your actual API key.
8. Run the application on your local Apache Tomcat server:
    - Right-click on the project in Eclipse.
    - Go to `Run As` -> `Run on Server`.
    - Select your configured Tomcat server and click `Finish`.
9. Access the WeatherApp through your web browser using the provided URL (usually `http://localhost:8080/WeatherApp`).

API Integration in Servlet:

- Created a Java servlet (MyServlet.java) to handle HTTP requests.
- In the doPost method, fetched the city name from the form input.
- Constructed the API URL with the city name and your API key (apiUrl) to fetch weather data.

HTTP Request to API:

- Used HttpURLConnection to establish a connection to the API endpoint.
- Set the request method to GET and retrieved the API response using input streams.

Processing API Response:

- The API response was in JSON format.
- Used the Gson library to parse the JSON response into a JsonObject.
- Extracted relevant weather data like temperature, humidity, wind speed, visibility, weather condition, and cloud cover from the JSON response.

Setting Request Attributes:

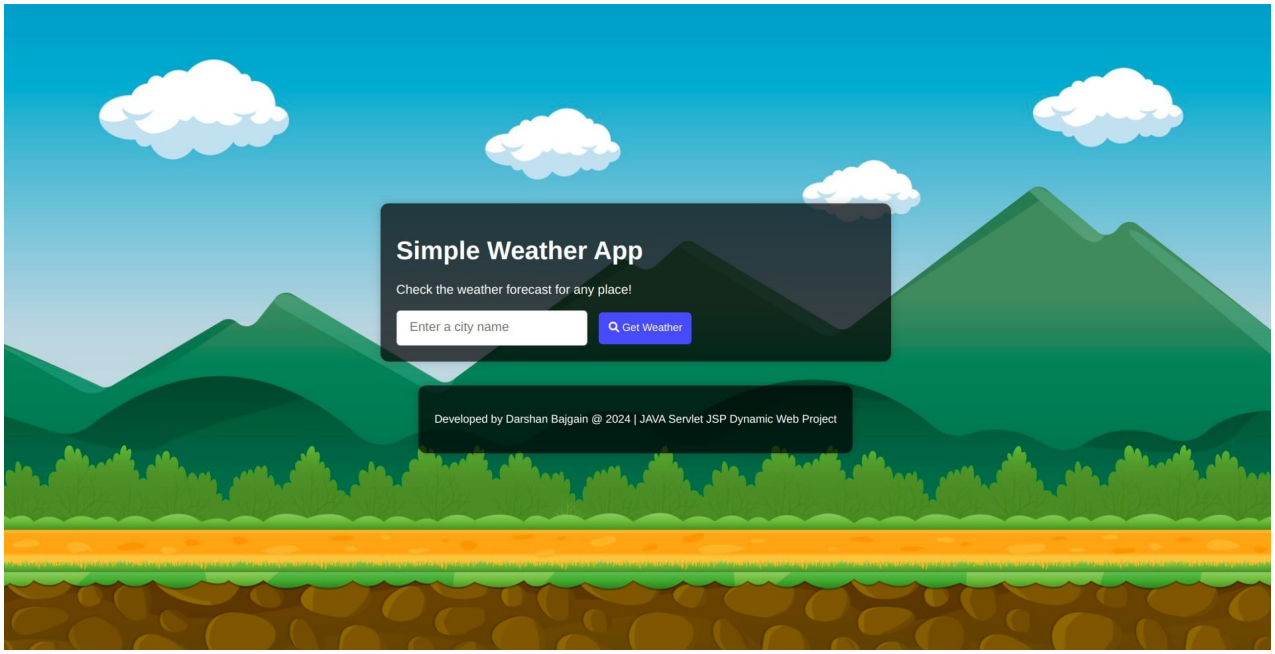
- Stored the extracted weather data, city name, date, time, and other relevant information as request attributes using HttpServletRequest.setAttribute().

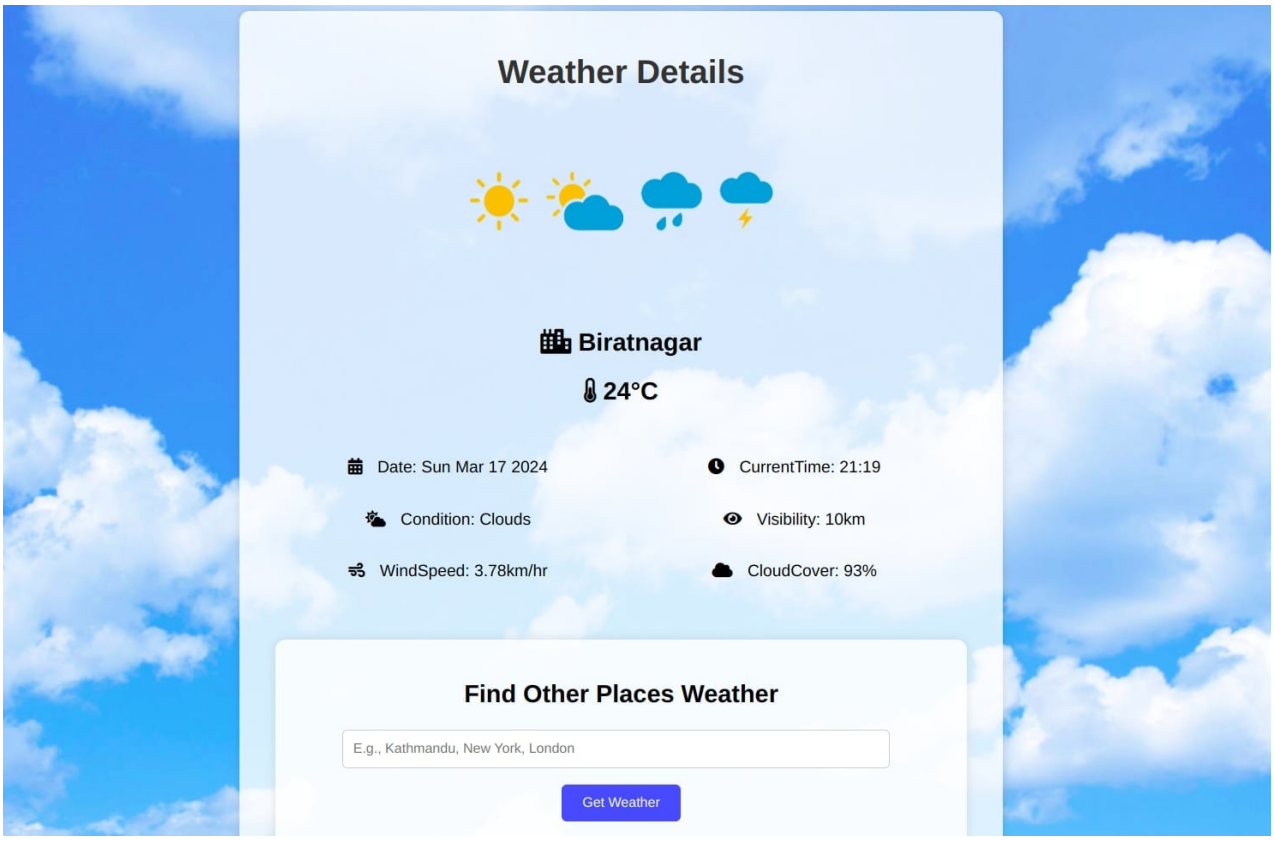
Forwarding Request to JSP:

- Forwarded the request to the JSP page (index.jsp) for rendering using RequestDispatcher.forward().

Displaying Data in JSP:

- In our JSP page (index.jsp), we used HTML and embedded Java code (EL expressions) to display the weather data.
- Accessed the data from request attributes using ${attributeName} syntax.





Let us look into the steps one by one.

Step 1: Team Formation Phase:

Team formation is a crucial step in any project it significantly impact on your project . In our project as we will be exploring about the web application for weather app so will will be going to require following skill sets.

1. **Front end Development (Html , CSS).**
2. **Back end Development (JavaScript).**
3. **Tester**
4. **UI/UX Developer**

Step 2: Creating Project Synopsys:

A project synopsis serves as a concise overview or summary of a proposed project, offering a brief but comprehensive insight into its objectives, scope, methodology, and expected outcomes. Let's create a Synopsys Report for Weather Application:

2.1 Introduction | Project Synopsys for Weather Forecasting Project :

Todays Weather app is a web application which will tell the users about the weather details of any particular city . The easy and Interactive User Interface will help our users to easily know about the temperature , wind speed , humidity and description about the weather .

2.1.1 Problem Statement:

Current weather apps lack simplicity and speed, making it challenging for users to quickly access accurate information for specific cities. There is a need for a streamlined web application that prioritizes user-friendly interfaces, delivering real-time, precise weather details for informed decision-making in travel, planning, and daily activities.

2.1.2 Proposed Solution for Weather Forecasting Project :

*To overcome this the problem , We are going to make an web application using **Html , CSS** and*

JavaScript *in which we will be providing user-friendly interface for easy navigation , Efficient weather searching , accurate and fast data collection.*

2.1.3 Objective of the Project:

The objective of the Todays weather project is to design and implement an efficient and user-friendly system that helps user to know about weather details of any city using its name only.

Primary Goals of the project:

- **User-friendly Interface**
- **Accurate weather Details**
- **Fast data Fetching**

2.2 Methodologies Used | Project Synopsys for Weather Forecasting Project :

In Weather App we are using various methodologies to solve our problems. Below are the detailed description about the technology used and methods we are applying in our project.

Technology Used:

Here we are developing a Weather application using ***HTML****,* ***CSS*** *for the frontend, and* ***JavaScript*** *for the backend involves a structured methodology. We are using OpenWeatherMap's* ***API*** *for the weather details .*

ER Model for Weather Forecasting Project :

An Entity-Relationship Diagram (ERD) for a Weather Application is the entities and their relationships within the system.

Entities:

- **User:** Attributes User Id (Primary Key)
- **City:** Attributes : City Name (Primary Key) , API Key value.
- **Weather Details:** Attributes: Temperature , Wind Speed , Weather Description , Humidity.

Relation:

- **Enters:** User enters the city name in the application.
- **Returns:** API returns an list of weather details having temperature , wind speed , humidity and weather details

Data Flow Diagram of Weather Forecasting Project :

Data Flow Diagram (DFD) serves as a visual representation of the flow of information within the system. This diagram illustrates how data, such as weather information, user details, and API Transactions in Weather Application.

Let's Draw a Dataflow Diagram for our project:

2.3 Features | Project Synopsys for Weather Forecasting Project :

The proposed Weather Application is designed to simplify the way to get weather details about any city . These are the priority features for our project:

- **User-friendly Interface**
  - Using Basic html, CSS and JS.
- **Accurate weather Details**
  - Temperature in that city
  - Wind speed in the city
  - Weather Description
  - Humidity
- **Fast data Fetching**
  - Using open weather map's api for it.

2.4 Limitations | Project Synopsys for Weather Forecasting Project :

Weather Application can offer many benefits, it may also have certain limitations. Here are some potential constraints associated with such a system:

1. Reliability: The accuracy of weather information heavily depends on the reliability of the data source. If the application relies on a single source, discrepancies or inaccuracies in that source can impact the reliability of the weather data.

2. Dependency on External API's: If the application relies on third-party APIs for weather data, it's subject to the availability and performance of those APIs. Downtime or changes to the external service can affect the functionality of the application.
3. City Coverage: The availability and accuracy of weather data may vary for different cities. Some locations may have more comprehensive and accurate data than others, potentially leading to incomplete or less reliable information for certain areas.
4. Device and Network Dependency: The user experience can be affected by the device's capabilities and network conditions. Slow internet connections or outdated devices may impact the responsiveness of the application.
5. Security and Privacy Concerns: Handling user data, such as location information, requires attention to privacy and security. Ensuring secure data transmission and storage is crucial to protect user information.

2.5 Future Scope | Project Synopsys for Weather Forecasting Project :
The future scope of a Our Weather Application developed using HTML, CSS, JS is promising, with opportunities for enhancement and expansion.
Some potential future avenues for the project include:
1. Integration of Advanced Technologies: Explore the integration of emerging technologies such as artificial intelligence (AI) and machine learning (ML) for intelligent future weather predictions using past dataset.
2. **Mobile Applications:** We can develop mobile applications for the same. As there are more mobile users than website users.
3. **Multi-language Support:** Expand the system's reach by incorporating multi-language support to cater to diverse user populations and potentially attract a global user base.
4. **Enhanced Security Measures:** Stay abreast of evolving cybersecurity threats and implement advanced security measures to safeguard user data 'location' and ensure the integrity of the system.
5. User Feedback Mechanisms: Strengthen user feedback mechanisms to continuously gather input on system performance, identify areas for improvement, and enhance user satisfaction.
Step 3: Requirement Gathering -
This is the next phase after the submission of the synopsis report. We can do this process before the Synopsys report creation as well , It is all depends upon the project and their requirements. Here after getting an overview about the project now we can easily do the requirement gathering for our project.
The Software Requirement Specification document of any project is the detailed description of the both functional and non-functional requirements proposed by the client.
Software Requirement Specification Document | Weather Forecasting Project
Below are some of the key points in a Software Requirement Specification Document:

- ***Introduction***
  - *Purpose*
  - *Scope*
- ***Overall Description***
  - *Product Perspective*
  - *System Interface*
  - *Product Function*
  - *Operating Environment*
  - *Assumptions and Dependencies*
- ***Functional Requirements***
  - *Software Requirements*
  - *Hardware Requirements*
  - *Database Requirements*
- ***Non-Functional Requirement***

- *Usability Requirements*
- *Security Requirements*
- *Availability Requirements*
- *Scalability Requirements*
- *Performance Requirements*

- ***Design***
  - *Control Flow Diagram*
  - *ER Model of LMS*
  - *Use Case Diagram*
- ***System Features***

Note: To know more about **What is a SRS Document** or **How to write a good SRS for your Project** follow these articles.

Let's Start building a Software Requirement Specification for Weather Application:

3.1 Introduction | Weather Forecasting Project (SRS):

3.1.1 Purpose:

The main objective of this document is to illustrate the requirements of the project **Weather Forecasting** . The document gives the detailed description of the both functional and non-functional requirements proposed by the client.

Developing a comprehensive weather forecasting application that provides accurate and real-time weather information to users. The app aims to offer a user-friendly interface with a focus on reliability, precision, and a wide range of features to enhance the overall user experience.

3.1.2 Scope:

A weather forecasting application is a software tool designed to provide users with up-to-date and accurate information about current and future weather conditions. These applications leverage data from meteorological sources, satellites, and weather stations to deliver forecasts, real-time weather updates, and other related information.

The primary goal of a weather forecasting app is to offer users the ability to plan their activities based on anticipated weather conditions.

3.2 Overall Description | Todays Weather App:

3.2.1 Product Perspective:

This app provides commercial weather forecasting services worldwide . it will utilize ideas passed by openweathermap and national weather services.

3.2.2 System Interface:

The user interface for the task will have a site that will have the live feed alongside the data about the climate. This site will utilize html , CSS , JavaScript and API requests for site.

3.2.3 Product Functions:

- It will show the current weather of the provided location.
- It will show the current wind speed of the location.
- It will also show the humidity level of the provided location.

3.2.4 Operating Environment:

Todays weather app is an web application which you can run using your favourite browser. In the application provided details and information is provided by the open weather map website which will help us for api data fetching.

3.2.5 Assumption and Dependencies:

Assumptions and dependencies play a crucial role in the development and operation of any application, including a weather forecasting web application. These factors influence the accuracy, reliability, and overall performance of the application.

Here are some key assumptions and dependencies for a weather forecasting web application:

Assumptions:

- Data Accuracy
- Stable Internet connection
- User Location Accuracy

- Data Source Availability

Dependencies:

- Api Integration
- Geolocation Services
- Server Infrastructure
- Data Storage
- Internet Connectivity

3.3 Functional Requirements | Todays Weather App:

3.3.1 Software Requirements:

This software package is developed using html , CSS for frontend and JavaScript for the backend. Using Vs Code as a text editor and Google Chrome for the execution of our code.
Required Specifications for our Device:

- **Operating System**: Windows 7, 8, 9, 10 .
- **Language**: Html , Css , Javascript.
- **API** : Openweathermap api

3.3.2 Hardware Requirements:

- **Processor**: Intel core i3 or above for a stable experience and fast retrieval of data.
- **Hard Disk**: 4GB and above
- **RAM**: 256 MB or more, recommended 2 GB for fast reading and writing capabilities which will result in better performance time.

3.4 Non-Functional Requirements:

3.4.1 Usability Requirements

- Our user interface should be interactive simple and easy to understand . The system should prompt for the user and administrator to login to the application for proper input criteria.
- Weather Application shall handle expected and non expected errors in ways that prevent loss in information and long downtime period.

3.4.2 Security Requirements

- System should be using secure web forecasting API.
- Normal users can just read information about the weather but they cannot edit or modify anything except their personal and some other information if required.
- System will have different types of users and every user has access constraints.
- Proper user authentication should be provided.
- Personal details like location should not be stored for security purpose.

3.4.3 Availability Requirements

Availability requirements for a weather application using APIs are crucial to ensure that the service is consistently accessible and operational.
Here are key availability requirements:

- **Uptime Percentage:** Maintain a high level of service availability, such as 99.9% uptime. Users rely on the weather application for real-time information, so a high uptime percentage ensures that the service is consistently accessible.
- **Load Balancing:** Use load balancing to distribute incoming traffic across multiple servers or instances. Load balancing helps distribute the load evenly, preventing individual servers from becoming overwhelmed and improving overall system performance and availability.
- **Monitoring and Alerting:** Implement continuous monitoring of key metrics and set up alerting systems to notify administrators of any issues or anomalies. Proactive monitoring allows for the rapid identification and resolution of potential problems, minimizing downtime.
- **Scalability:** Design the system to be scalable, allowing for the seamless addition of resources during periods of increased demand. Scalability ensures that the application can handle varying levels of traffic and user activity without degradation in performance.
- **Backup and Recovery:** Regularly back up critical data and implement robust recovery procedures. In the event of data loss or system failures, a well-defined backup and

recovery strategy ensures that the application can be restored quickly and efficiently.

3.4.4 Performance Requirements:

- The system shall accommodate high number of users simultaneously and users can check the weather of any location any number of times.
- Responses to view information shall take no longer than 5 seconds to appear on the screen.

4.5.4 Error Requirements:

Weather app shall handle expected and non-expected errors in ways that prevent loss in information and long downtime period.

An alert should be shown if the API is not working properly which will improve the uptime of the service.

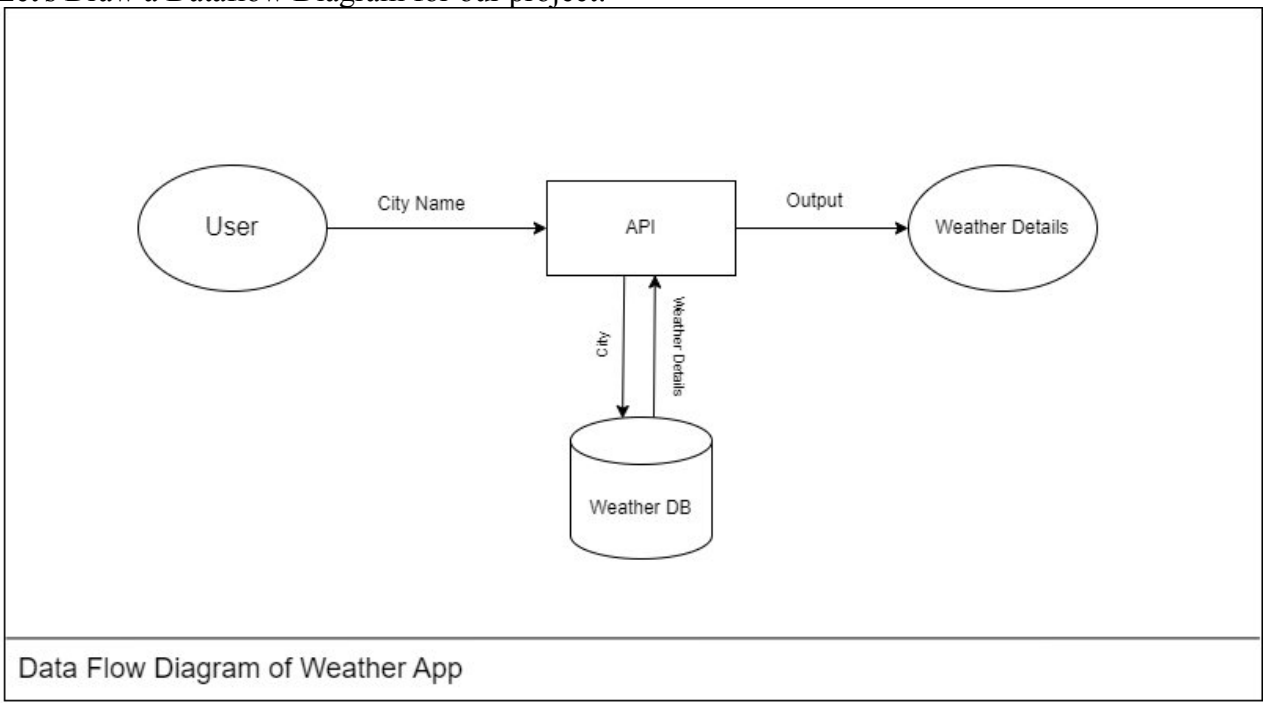
3.5 Design:

The design phase in weather forecasting app development is a crucial stage where the conceptual ideas and requirements are transformed into a detailed and visually appealing blueprint. This phase involves creating the Data flow Diagrams, ER model design, and the overall architecture of the weather application

3.5.1 Data Flow Diagram of Weather Forecasting Project :

Data Flow Diagram (DFD) serves as a visual representation of the flow of information within the system. This diagram illustrates how data, such as weather information, user details, and API Transactions in Weather Application.

Let's Draw a Dataflow Diagram for our project:

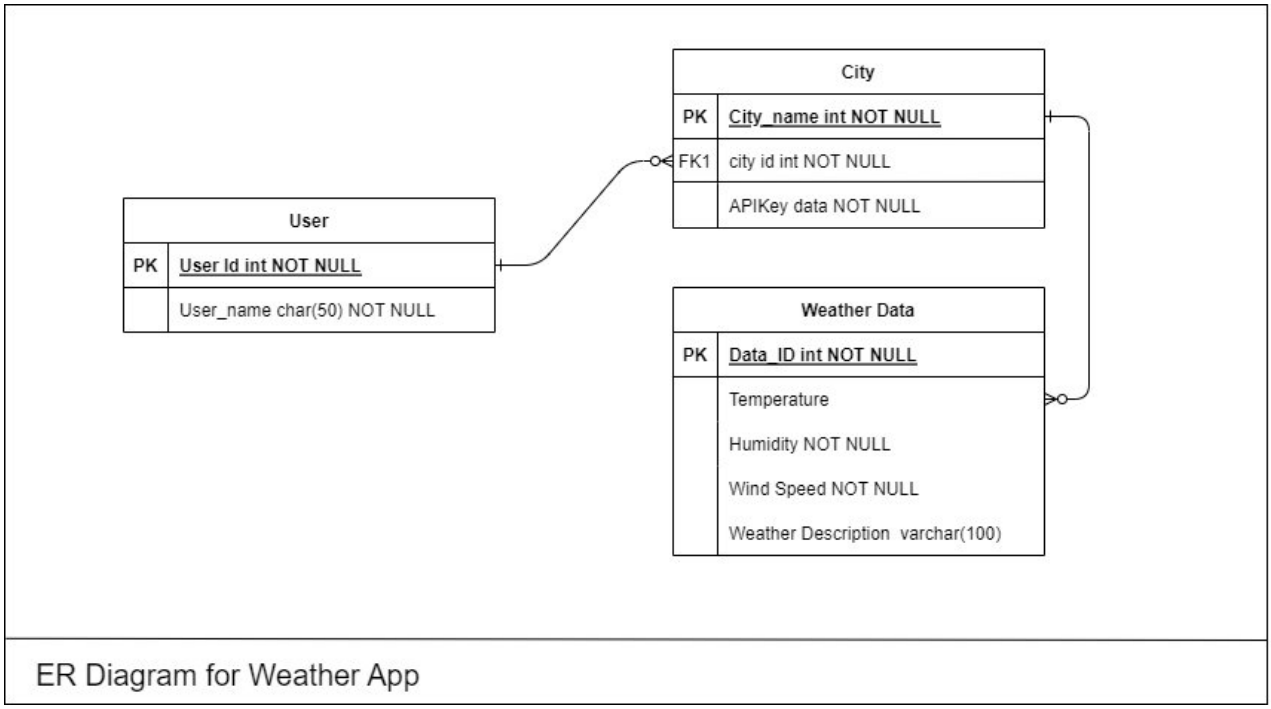


Data Flow Diagram for Weather Forecasting Project

3.5.2 ER Model for Weather Forecasting Project :

An Entity-Relationship Diagram (ERD) for a Weather Application is the entities and their relationships within the system.

Let's Draw an ER Diagram for our Weather Application:



ER Diagram for Todays Weather App

Entities:

- **User:** Attributes User Id (Primary Key)
- **City:** Attributes : City Name (Primary Key) , API Key value.
- **Weather Details:** Attributes: Temperature , Wind Speed , Weather Description , Humidity.

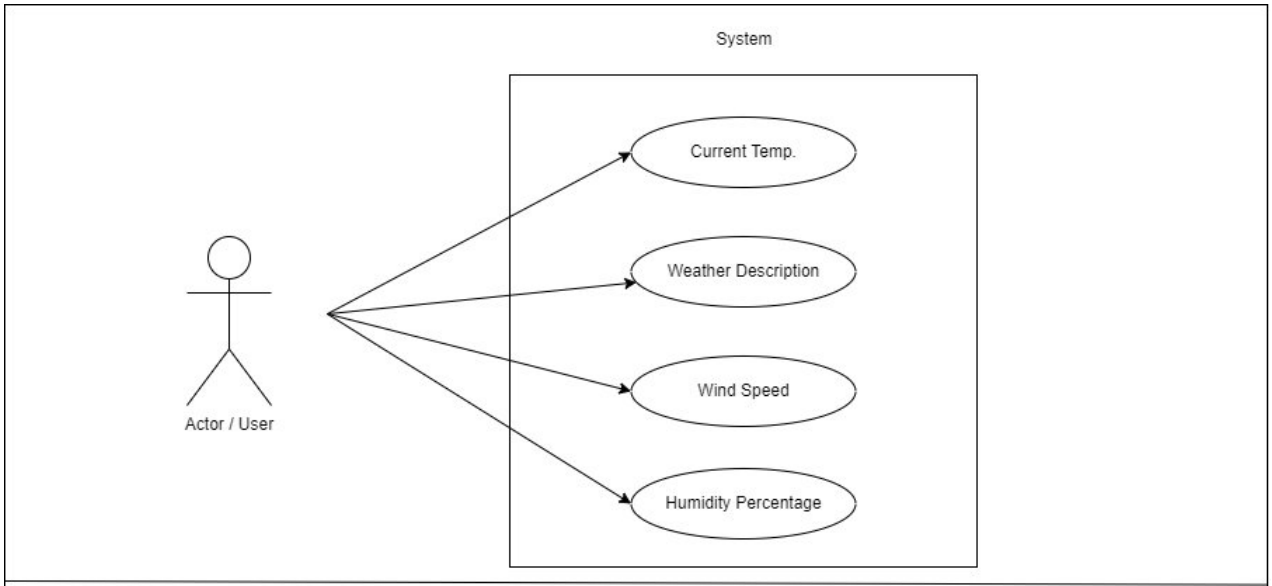
Relation:

- **Enters:** User enters the city name in the application.
- **Returns:** API returns an list of weather details having temperature , wind speed , humidity and weather details

3.5.3 Use Case Diagram:

A use case diagram is a visual representation of the functional requirements of a system, illustrating how users interact with the system and the system's responses. For a weather forecasting app, the use case diagram will include actors such as "User," "Weather Data Provider," or "Administrator".

Here's a simplified use case diagram for a weather forecasting app:



Use case Diagram for Weather Forecasting App

Description of the Use Case Diagram:

- **View Current Weather:**
  - **Actor:** User
  - **Description:** Allows the user to view the current weather conditions for their selected location.
- **View Hourly Weather Description:**
  - **Actor:** User
  - **Description:** Enables the user to check the weather description for the selected location.
- **View Daily Forecast:**
  - **Actor:** User
  - **Description:** Permits the user to access the daily weather forecast for the chosen location.
- **Set Location Preferences:**
  - **Actor:** User
  - **Description:** Allows the user to set and manage location preferences for weather forecasts.
- **Receive Weather Alerts:**
  - **Actor:** User
  - **Description:** Enables the user to receive alerts for severe weather conditions or customized weather events.

Step 4: Coding or Implementation of Weather Forecasting Project:

At this stage, the fundamental development of the product starts. For this, developers use a specific programming code as per the design. Hence, it is important for the coders to follow the protocols set by the association. Conventional programming tools like compilers, interpreters, debuggers, etc. are also put into use at this stage.

In our project we are using html , CSS and JavaScript to build the project so in this stage we are going to code our project. Before going further lets talk about the environment we need for the project.

4.1 Environment Creation:

In this stage we are going to create the environment to build our project, We will install all required software and extensions for ease in the coding part.

Required Softwares:

- **VsCode:** Vs Code is a widely used text editor for development purpose .

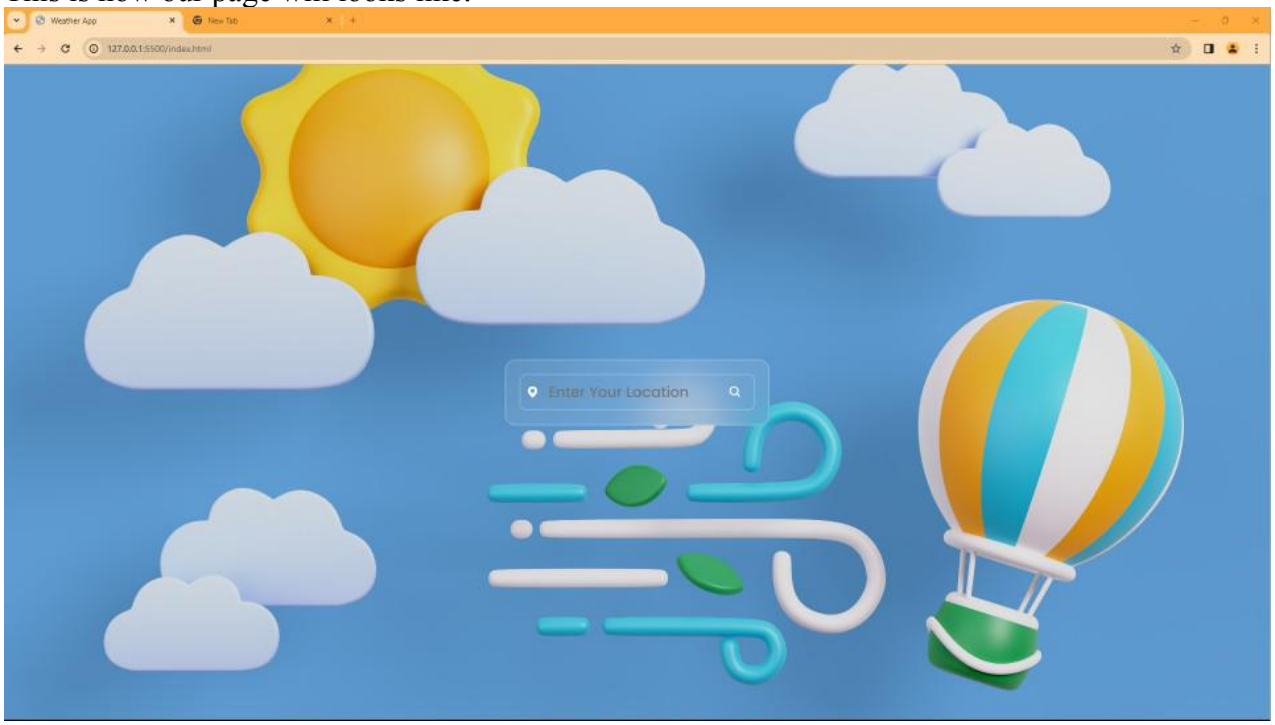
- **Google Chrome:** You need to install a web browser to execute the html code. You can use any of your favourite web browser.

Extensions :

- **Live Server**: You can use live server extension because It enables you to right-click an HTML document, and it runs a server for you and opens a browser window with the file in it.

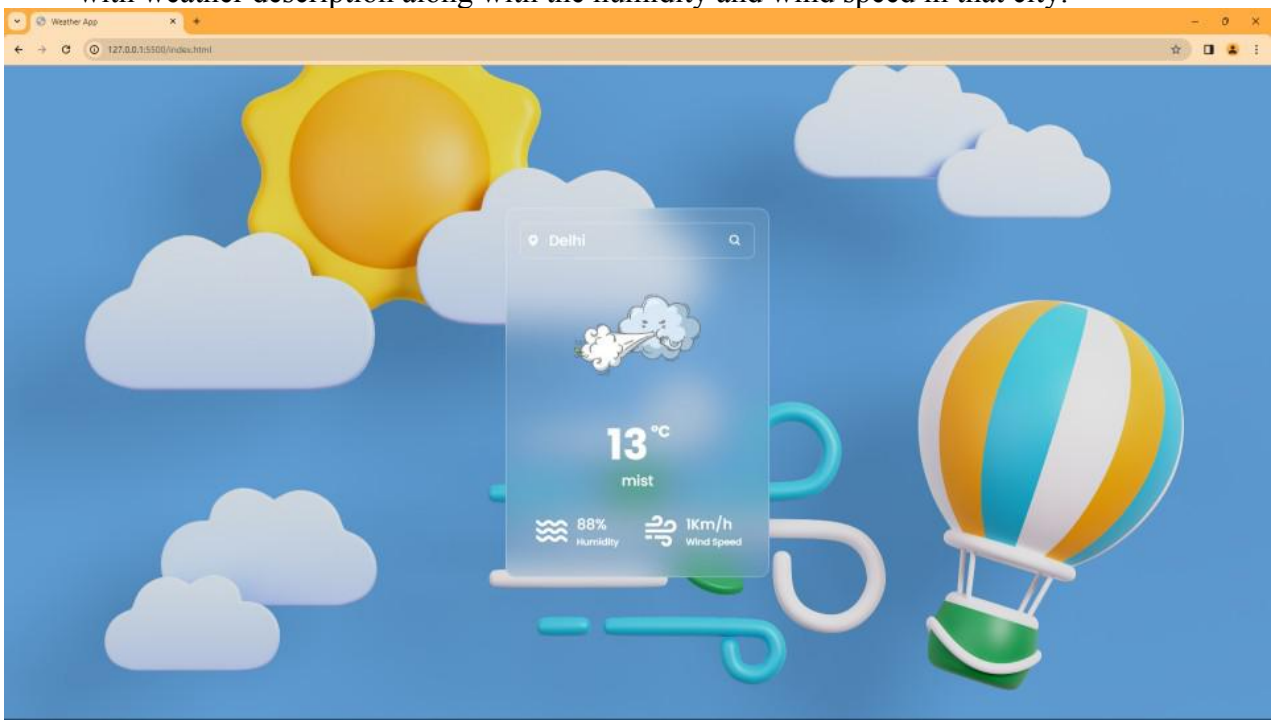
4.2 Implementation of Weather Forecast Application | Frontend Development:

In this stage we are going to develop our frontend part of the project .
This is how our page will looks like:



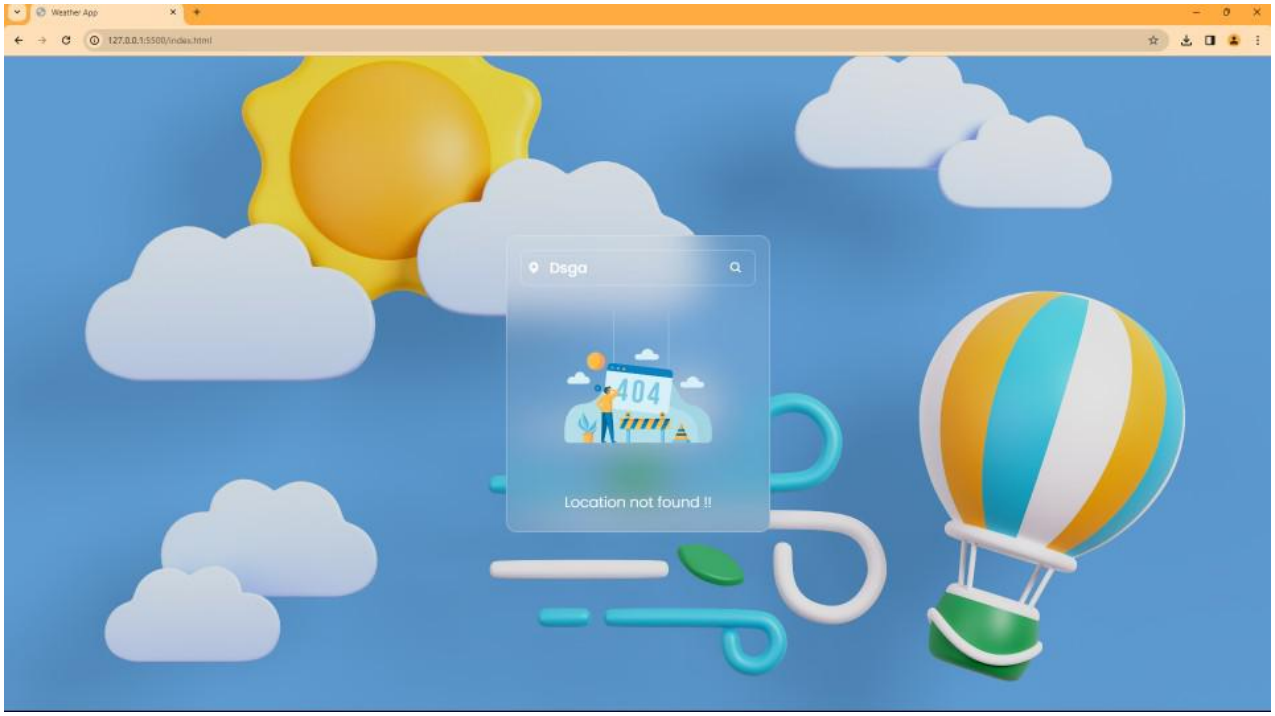
Home Page for Weather Forecasting Project

- After providing the city value by user the app will show us the Temperature of the city with weather description along with the humidity and wind speed in that city.



Weather Forecasting Project

- If user provide the any invalid data in the search field then our page will respond with an 404 error.



Functionalities of this page:

- User can put the City name and get the weather details:
  - Temperature
  - Humidity
  - Wind Speed
  - Weather Description
- Easy User Interface

Code:

Below is the Code for Creating above page:

```
const container = document.querySelector('.container');
const search = document.querySelector('.search-box button');
const weatherBox = document.querySelector('.weather-box');
const weatherDetails = document.querySelector('.weather-details');
const error404 = document.querySelector('.not-found');

search.addEventListener('click',() =>{
    // Enter your API Key in the APIKey variable
    // You can use any weather api for the project
    // Here we are using openweathermap's API which
    // you can find in their website by searching
    // weather API.
    const APIKey = '';
    const city = document.getElementById('search-btn').value;
    if(city==''){
        return ;
    }


fetch(`https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid
=${APIKey}`).then(response => response.json()).then(json => {
```

```
    if(json.cod == '404'){
        container.style.height = '450px';
        weatherBox.classList.remove('active');
        weatherDetails.classList.remove('active');
        error404.classList.add('active');
        return;
    }

    container.style.height = '560px';
    weatherBox.classList.add('active');
    weatherDetails.classList.add('active');
    error404.classList.remove('active');

    const image = document.querySelector('.weather-box img');
    const temperature = document.querySelector('.weather-box .temperature');
    const description = document.querySelector('.weather-box .description');
    const humidity = document.querySelector('.weather-details .humidity span');
    const wind = document.querySelector('.weather-details .wind span');

        switch(json.weather[0].main){
            case 'Clear':
                image.src = 'images/clear-new.png';
                break;
            case 'Rain':
                image.src = 'images/rain-new.png';
                break;
            case 'Snow':
                image.src = 'images/snow-new.png';
                break;
            case 'Clouds':
                image.src = 'images/cloud-new.png';
                break;
            case 'Mist':
                image.src = 'images/mist-new.png';
                break;
            case 'Haze':
                image.src = 'images/mist-new.png';
                break;
            default:
                image.src = 'images/clear-new.png';
        }
        temperature.innerHTML = `${parseInt(json.main.temp)}<span>°C</span>`;
        description.innerHTML = `${json.weather[0].description}`;
        humidity.innerHTML = `${json.main.humidity}%`;
        wind.innerHTML = `${parseInt(json.wind.speed)}Km/h`;
    });

});
```

Step 5: Testing of Weather Forecasting Application:

Testing is a crucial phase in the development of a ***weather forecasting project*** *to ensure that it*

meets its intended requirements, functions correctly, and is free of bugs.
Below are some key steps and considerations for the testing phase of a weather forecasting application:

1. **Unit Testing:**
    - Test individual modules or components of the system in isolation to ensure they function as intended.
    - We have a major weather details module which uses api to fetch the weather data, in this testing step we take we make sure proper functionality of each component.
2. **Integration Testing:**
    - Verify that different modules and components of the Todays weather application work together seamlessly.
    - Test data flow and interactions between various parts of the system.
3. **Functional Testing:**
    - Validate that the weather forecasting details its intended functions accurately and efficiently.
4. **User Interface (UI) Testing:**
    - Ensure that the user interface is user-friendly, intuitive, and visually appealing.
    - Check for consistency in design elements and responsiveness across different devices.
5. **Performance Testing:**
    - Assess the system's performance under normal and peak load conditions.
    - Check response times, scalability, and overall system stability.
6. **Security Testing:**
    - Identify and rectify any security vulnerabilities in the system.
    - Ensure that user data is handled securely, and unauthorized access is prevented specially in case of location.

Step 6- Creating Project Presentation on Weather forecasting Application:

In this phase of software development, Team will have to present their work in front of authorities and they will judge your work and give suggestions on the improvement areas.
The ideal length of the ppt should be min 10 slides and maximum 15 slides , you will not have too much time to explain your project so prepare your presentation carefully using important key points.
Some of the key points (slides) which your presentation should have are given below:

1. **Project Name and Team Details**
2. **Introduction**
3. **Purpose**
4. **Project Scope**
5. **Problem Statement**
6. **Proposed Solution**
7. **Product Functionalities**
8. **Flow chart of the project**
9. **Analysis of model**
10. **Summary**

Future Enhancements for Weather forecasting Application:

- Integration of future weather prediction using Machine learning technologies.
- Integration of any natural disaster prediction in the location.
- Integration of local time and international time in that location, we can add this using any api.
- We can add top weather headlines in the nearby locations as well.
- We can add last 10 days time line of weather forecasting of the selected location as well.