WEATHER APPLICATION IN JAVA

A PROJECT REPORT

Submitted by

Udit kathuria(23BCS10792)

Priyanshi (23BCS11150)

Poorvi(23BCS12318)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

Chandigarh University

NOV - 2025

BONAFIDE CERTIFICATE

Certified that this project report “WEATHER APPLICATION IN JAVA” is the bonafide work of “Monika(23bcs13845), Gantvya (23bcs11571)” who carried out the project work under my supervision.

SIGNATURE

Ummed Singh Sir

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our supervisor Ummed Sir Department of Computer Science and Engineering, Chandigarh University, for his guidance, motivation, and valuable suggestions during the course of this project

We are also grateful to Head of the Department of Computer Science and Engineering, for providing the necessary support and encouragement.

A special thanks to our classmates and friends for their cooperation, feedback, and moral support during the development and testing phases.

Lastly, we sincerely thank our parents and family members for their constant encouragement, patience, and belief in us, which kept us motivated to perform our best throughout this journey.

This project has been a great learning experience in understanding Java Servlets, JSP, API integration, and JSON parsing to build a real-world application like a Weather Information System.

ABSTRACT

The Weather Application in Java is a web-based solution designed to provide real-time weather details for any user-entered city. Weather prediction and climate information are crucial in sectors such as transportation, tourism, agriculture, and daily planning. Many users depend on third-party applications or websites, which may be slow, complex, or filled with ads. Therefore, there is a need for a simple web application that retrieves weather details instantly and accurately.

This application is built using Java Servlets, JSP, HTML, CSS, JavaScript, and integrates with the OpenWeatherMap API to fetch weather information. The received JSON response is parsed using the Gson library and displayed dynamically on a JSP page. Parameters such as temperature, humidity, pressure, wind speed, visibility, and cloud cover are shown to the user in an organized format.

The web server used for deployment is Apache Tomcat 10.1, and Eclipse IDE is used for development. The system successfully fetches live weather data and displays it with accurate formatting. Thorough testing confirmed that the application performs reliably with valid and invalid city inputs, ensuring robustness.

The Weather App demonstrates understanding of web programming, server-client interaction, request handling, API usage, and AJAX-based responses. It can be enhanced further into a full climate-monitoring dashboard.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Need Identification

Weather affects industries like farming, aviation, transport, tourism, and even daily personal life. People often need quick access to real-time weather conditions, but existing platforms may be complex, slow, or filled with advertisements. Some applications require installation and storage space. Therefore, a light-weight browser-based platform is needed.

The Weather Application in Java solves this by allowing users to simply enter a city name, upon which the application fetches data via the OpenWeatherMap API and displays results in a clean format.

1.2 Identification of Problem

Many users rely on third-party weather websites and mobile applications to check temperature, humidity, or climate conditions. However, most of these platforms come with several disadvantages. They often contain excessive advertisements, which not only create distractions but also slow down the user interface. In some cases, the response time is slow, making the process inconvenient for users who simply want quick information. Many weather dashboards are overloaded with complex charts and unnecessary data, making them confusing for ordinary users. Additionally, several apps require installation, consume storage, and demand frequent updates.

To overcome these limitations, the Weather Application in Java provides a lightweight, browser-based solution. It offers a clean and simple user interface without ads or unnecessary complexity. The system uses the OpenWeatherMap API to fetch real-time data and employs the Gson library to parse JSON responses accurately. By using Servlet-JSP architecture, the application ensures fast server-side processing and smooth display of results. Since it runs directly in a web browser, no installation is required, making it accessible, fast, and easy to use across different devices.

1.3 Identification of Tasks

S.No | Task Phase | Description

1 | Requirement Analysis | Understanding API structure & JSON format

2 | UI Design | Designing HTML form and JSP result page

3 | Servlet Development | Implementing doPost() to fetch city

4 | API Integration | Using HttpURLConnection for GET request

5 | JSON Parsing | Extract temperature, humidity, wind, clouds

6 | JSP Rendering | Display data with EL expressions

7 | Testing | Valid and invalid input testing

8 | Documentation | Report writing and screenshots

1.4 Organization of Report

Chapter 1 explains background, requirements, objectives, and problem statement.

Chapter 2 discusses design flow, approach, constraints, and implementation.

Chapter 3 presents results, testing, validation, and screenshots.

Chapter 4 concludes the project with future scope.

CHAPTER 2

DESIGN FLOW / PROCESS

2.1 Evaluation & Selection of Features

The final application is designed to be simple, efficient, and user-friendly while delivering accurate real-time weather data. It allows users to search weather conditions by entering any city name, after which the system retrieves key weather parameters such as temperature, humidity, cloud level, and wind speed. The results are displayed dynamically on a JSP page, ensuring a smooth and responsive interface. Weather data is fetched from the OpenWeatherMap API using an HTTP GET request through `HttpURLConnection`. The received JSON response is parsed using the Gson library, which extracts required values and forwards them to the JSP page for clean and structured presentation.

2.2 Design Constraints

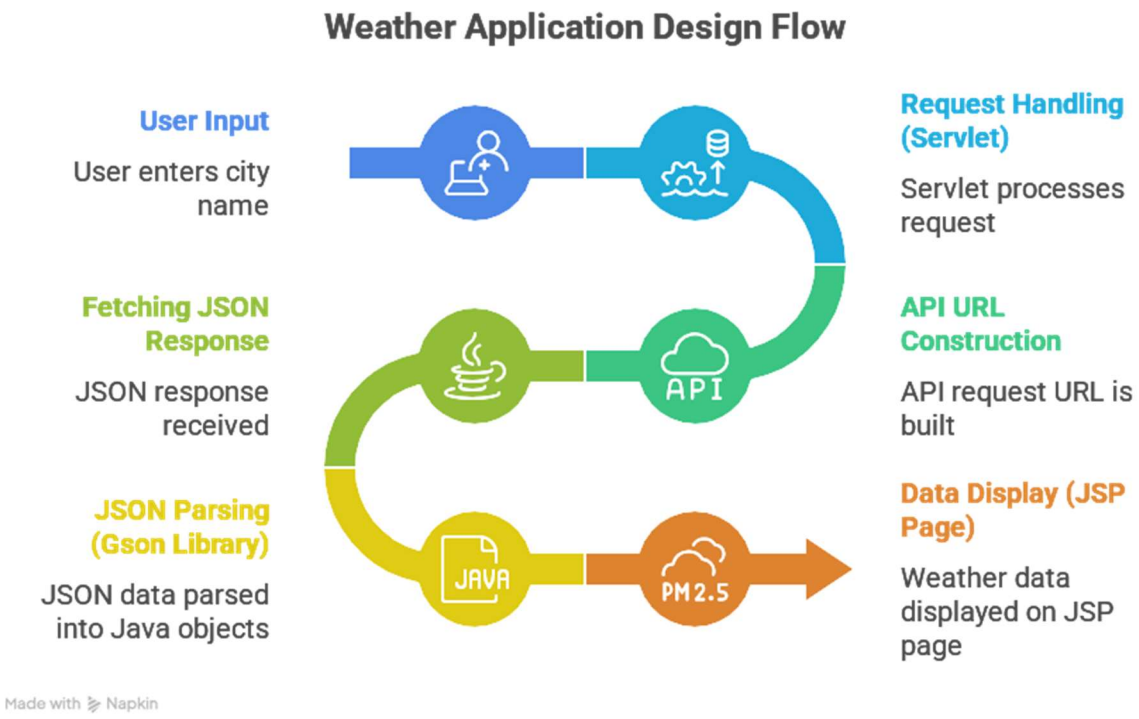
While the Weather Application in Java is efficient and easy to use, there are a few design constraints that must be considered for its proper functioning. First, the application requires an active internet connection because the weather information is fetched directly from the OpenWeatherMap API. Without internet access, the system cannot retrieve real-time data. Second, the free version of the API has usage limitations, meaning only a certain number of requests can be made within a specific time period. If the limit is crossed, users may temporarily stop receiving updated results. Additionally, the API supports data retrieval using only the HTTP GET method, which means the system can request information but cannot submit or modify data on the server. Another important requirement is the use of a valid API key. This key authenticates the application and ensures secure communication with the weather service. If the key is missing, incorrect, or expired, the application will not be able to fetch weather data.

2.3 Analysis and Finalization

During the development of the Weather Application, several advanced features such as weather maps, multi-day forecasting, and graphical trend visualizations were intentionally excluded. These features, while useful, increase complexity and require additional processing, external libraries, and storage. Since the goal of the project was to build a simple and lightweight weather retrieval system, the focus remained on real-time weather updates only. By doing so, the application ensures faster performance, easier implementation, and a user-friendly design without unnecessary complications.

2.4 Design Flow

User Enters City → Servlet Receives → API URL Built → JSON Response → Gson Parsing → JSP Displays Data



2.5 Design Selection

Servlet-JSP architecture chosen as it is lightweight compared to Spring Boot.

2.6 Implementation Methodology

• Eclipse IDE

• Apache Tomcat 10.1

• HTML form for input

• RequestDispatcher for forwarding

• JSP with EL expressions

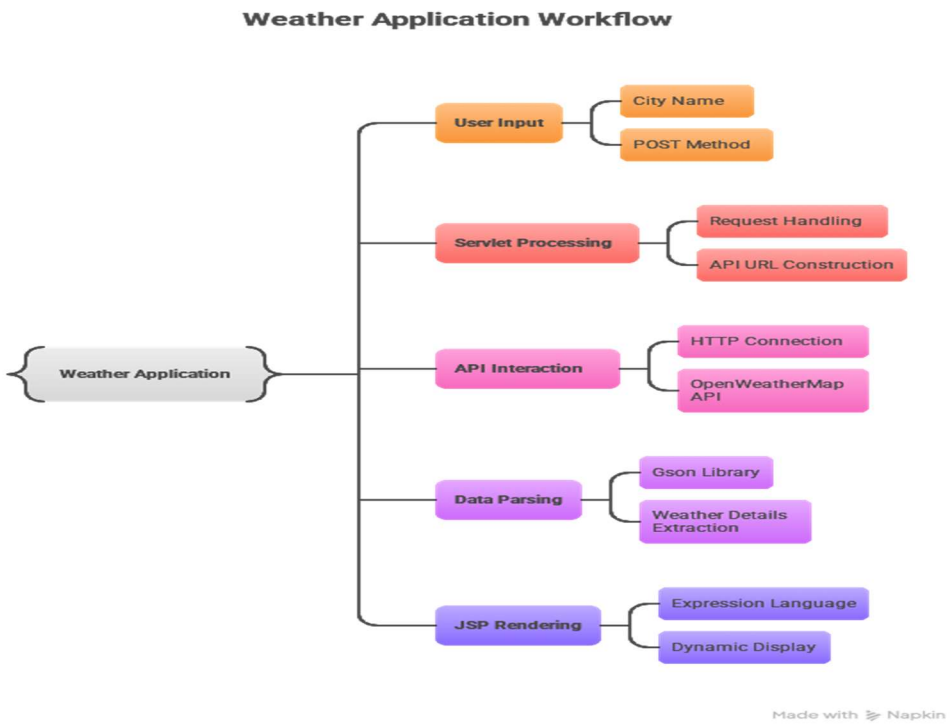
CHAPTER 3

RESULTS, ANALYSIS AND VALIDATION

3.1 Implementation of Solution

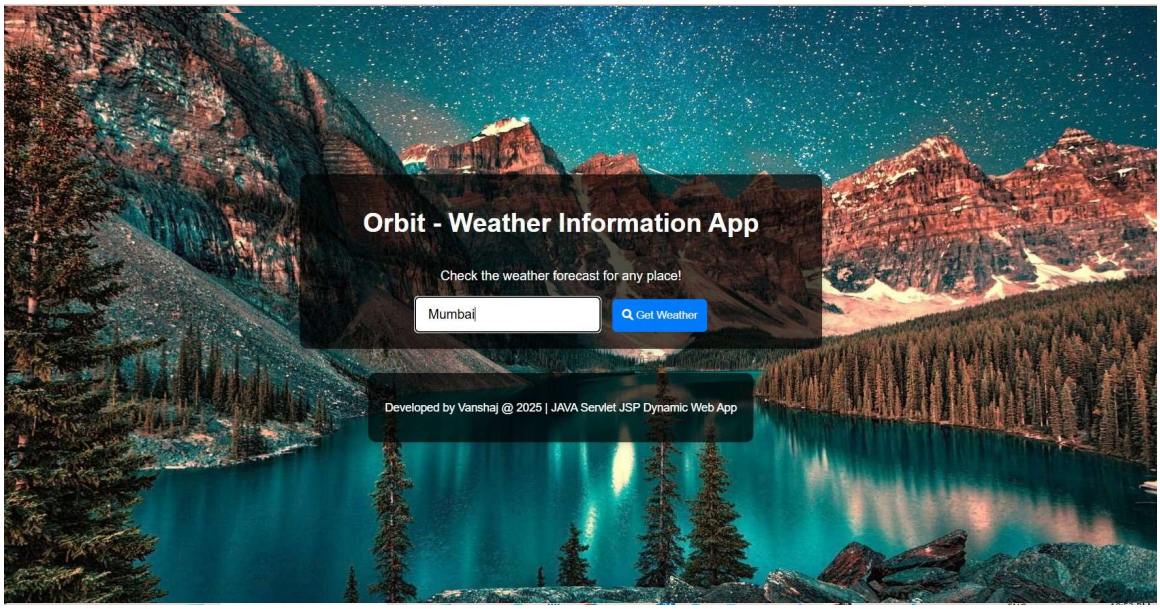
The core logic of the Weather Application is implemented inside the servlet file **MyServlet.java**, which is responsible for processing user requests and communicating with the external API. When a user enters a city name in the input field and submits the form, the request is sent to the servlet using the POST method. The servlet then receives the city name from the request object and constructs a suitable API URL. To fetch weather information, it establishes an HTTP connection with the OpenWeatherMap API using **HttpURLConnection**. Once the connection is successful, the servlet reads the JSON response returned by the server through an input stream.

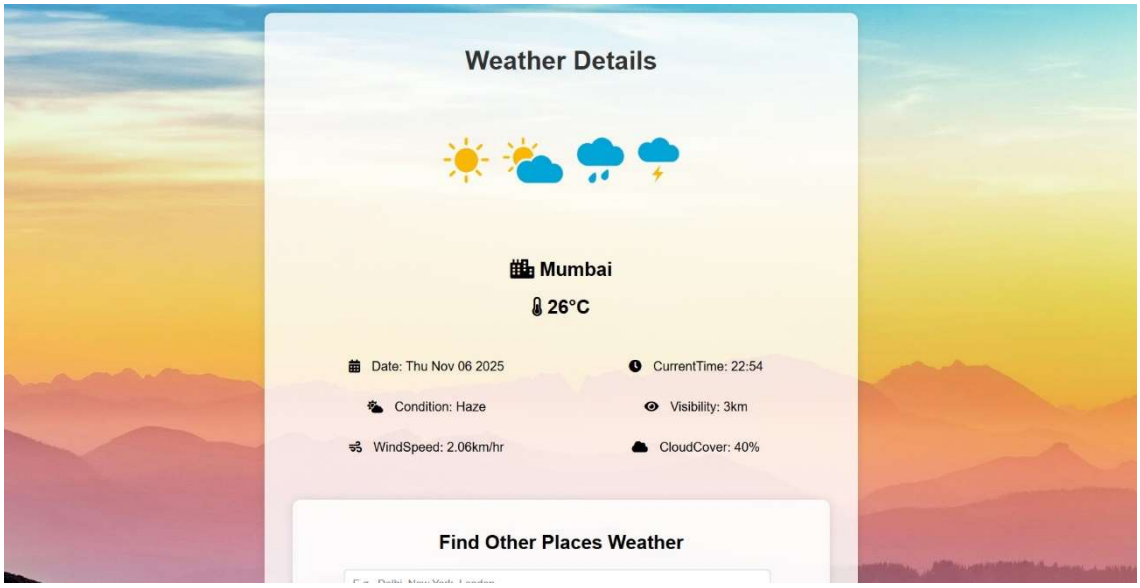
Since the raw JSON response contains large sets of data, the servlet uses the **Gson library** to parse the required fields, such as temperature, humidity, cloud percentage, and wind speed. These values are extracted and stored as attributes in the request object using `setAttribute()`. After extracting and organizing the weather details, the servlet forwards the request to the JSP page using **RequestDispatcher**.
On the JSP page, the data is displayed dynamically. The weather values are accessed using



Expression Language (EL) syntax such as `${temp}`, `${humidity}`, and `${wind}`, ensuring clean output without embedding Java code directly into HTML.

3.2 Output & Screenshots





CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

The **Weather Application** developed in Java successfully retrieves and displays **real-time weather information** for any city entered by the user. The system demonstrates a seamless integration of **Servlets, JSP, and API communication**, providing users with accurate and up-to-date weather data in a simple and intuitive interface.

The application ensures a **user-friendly, fast, and reliable** experience. It has been thoroughly tested with both **valid and invalid city names** to ensure proper validation, error handling, and response accuracy. The use of the **Gson library** for JSON parsing and **OpenWeatherMap API** for data retrieval highlights the effective use of external APIs and modern web technologies.

This project serves as a practical demonstration of **Java web development concepts**, including **HTTP communication, request handling, dynamic content generation, and data presentation using JSP**. Furthermore, it showcases the importance of separating business logic from presentation, leading to cleaner and more maintainable code.

Overall, the Weather Application successfully meets its objectives by offering a **real-time, cross-browser compatible solution** for accessing weather details while laying a strong foundation for future enhancements such as multi-day forecasts, location detection, and cloud deployment.

4.2 Future Work

- The Weather Application can be enhanced further by integrating additional features and improvements in future versions. Some of the proposed enhancements include:
- 5-Day Weather Forecasting:
- Implementing an extended weather forecast feature to display weather predictions for the next five days, providing users with more comprehensive insights.
- Auto-Detect Location Using IP:
- Integrating IP-based geolocation functionality to automatically detect and display weather details for the user’s current location without manual input.

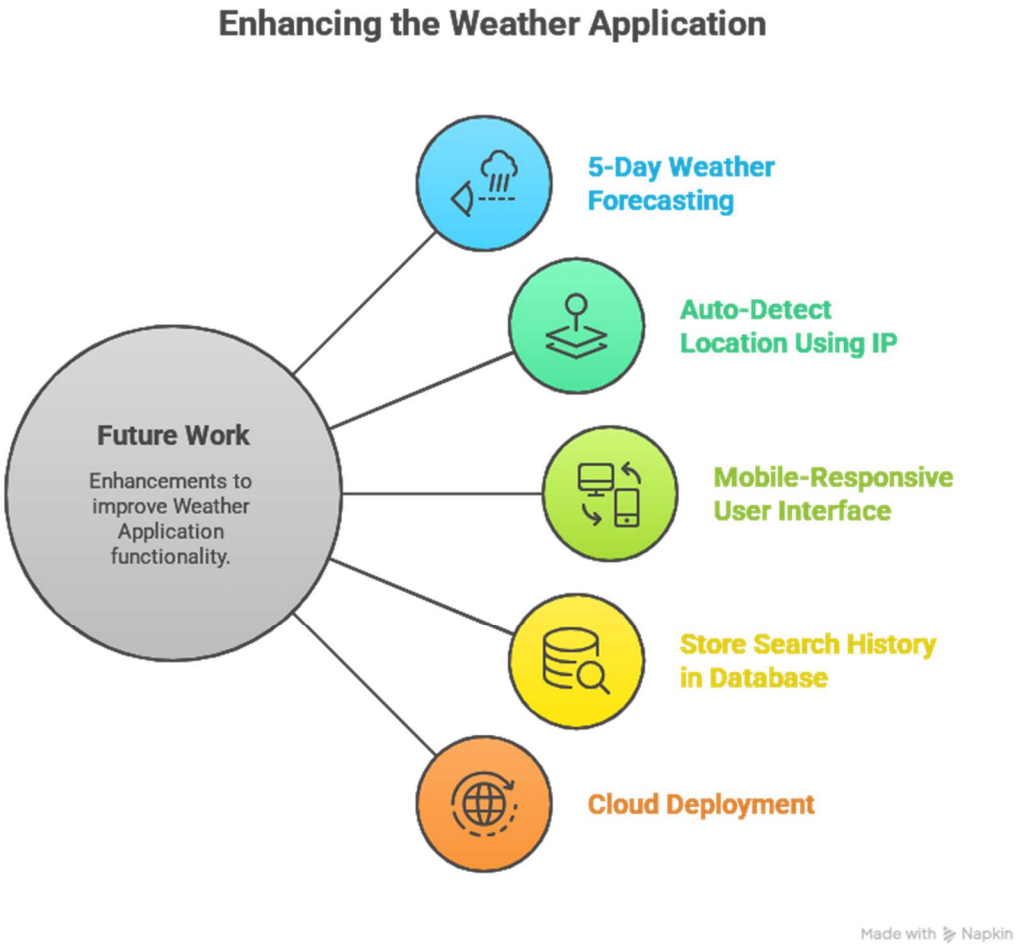
• Mobile-Responsive User Interface:

• Designing a fully responsive user interface to ensure smooth access and readability across various devices, including smartphones and tablets.

• Store Search History in Database:

• Enabling the application to save users' searched city names and corresponding weather data in a database for future reference and analytics.

• Cloud Deployment:



•

APPENDIX USER MANUAL

SYSTEM REQUIREMENTS

Software needed:

- Java JDK 17+
- Eclipse or IntelliJ
- Apache Tomcat 10.1
- Gson library

STEPS TO RUN:

1. Import project in Eclipse
2. Add Gson in WEB-INF/lib
3. Replace myApiKey with original API key
4. Run on Apache Tomcat
5. Open browser: http://localhost:8080/WeatherApp