PARTICLE EXPLOSION SIMULATION

A PROJECT REPORT

Submitted by

JASLEEN KAUR SOHAL	**(22BCS13216)**
ANKUR SINGH NEGI	**(22BCS15346)**
HARSHAD FOZDAR	**(22BCS10263)**

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

APRIL 2025

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

BONAFIDE CERTIFICATE

Certified that this project report “Weather Information App” is the bonafide work of “Jasleen Kaur Sohal”, “Ankur Singh Negi”, and “Harshad Fozdar” who carried out the project work under my supervision.

SIGNATURE

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 IDENTIFICATION OF CLIENT

With rising climate awareness and dependence on real-time weather data, providing timely and accessible weather updates has become essential. Clients for this system include students, commuters, event organizers, and professionals relying on accurate forecasts. The system targets any user with internet access seeking weather insights via a web application..

1.2 IDENTIFICATION OF PROBLEM

Despite multiple weather websites, a user-friendly, Java-based educational app that demonstrates the use of Servlets, JSP, and XML is lacking. Developers and learners find limited open-source examples that integrate these technologies for real-time weather access..

1.3 IDENTIFICATION OF TASKS

- Researching API-based weather data retrieval using XML
- Designing a web interface with JSP for user inputs
- Developing Servlets to process user requests and fetch XML data
- Parsing XML to extract temperature, humidity, and conditions
- Displaying data in a readable format with validation
- Testing application responsiveness and accurac

1.4 TIMELINE

- Week 1: Requirement analysis, API research
- Week 2: JSP design and front-end creation
- Week 3: Servlet setup and XML handling
- Week 4: Integration of frontend, backend, and data parsing
- Week 5: Testing, bug fixes, and enhancements
- Week 6: Report writing and submission preparation

1.5 ORGANIZATION OF THE REPORT

Chapter 1 covers the project overview and objectives.
Chapter 2 discusses the related research and background.
Chapter 3 explains the design and development process.
Chapter 4 shows the implementation and outcomes.
Chapter 5 concludes the project with future improvement scopes.

CHAPTER 2
BACKGROUND STUDY

2.1 TIMELINE OF THE REPORTED PROBLEM

Weather apps have existed for over a decade, but with technological shifts, the need to integrate traditional Java web technologies with real-time weather APIs has grown. Legacy applications often rely on outdated data or formats that are no longer user-friendly.

2.2 PROPOSED SOLUTIONS

By using open APIs such as OpenWeatherMap, this application fetches XML-formatted weather data which is parsed via Java Servlets. The results are dynamically displayed via JSP, allowing separation of concerns, easier maintenance, and scalability.

2.3 BIBLIOMETRIC ANALYSIS

Sources such as IEEE, W3Schools, and Java tutorials suggest increased interest in XML-based web services. However, examples that use all three — Java Servlets, JSP, and XML — are less frequent, reinforcing the need for practical implementations.

2.4 REVIEW SUMMARY

There's a lack of user-oriented, Java-centric web apps for weather data. Existing examples either avoid XML or bypass JSP, which leaves gaps in full-stack Java training. Our solution bridges this gap.

2.5 PROBLEM DEFINITION

To build a real-time Weather Information App that:

- Accepts city/location input via JSP
- Uses Servlets to make an HTTP request to a weather API
- Parses returned XML data for temperature, condition, and humidity
- Displays output in a user-friendly JSP interface
- Ensures modularity and efficient data parsing.

2.6 GOALS/OBJECTIVES

- Create an interactive JSP-based frontend
- Retrieve weather data in XML format
- Parse and display useful weather metrics
- Ensure efficient response time and high accuracy
- Maintain clean code architecture separating logic and view

CHAPTER 3

DESIGN FLOW/ PROCESS

3.1 EVALUATION & SELECTION OF SPECIFICATIONS/FEATURES

The application uses:

- Java Servlet for request handling
- JSP for user interface
- DOM/SAX parser for XML parsing
- OpenWeatherMap API (or similar) for data
- Apache Tomcat server for deployment

3.2 DESIGN CONSTRAINTS

- Network connectivity is required for real-time data
- API rate-limits affect frequency of queries
- XML data structure must be interpreted correctly
- Servlet lifecycle and exception handling must be managed

3.3 ANALYSIS OF FEATURES AND FINALIZATION SUBJECT TO CONSTRAINTS

Finalized features:

- Input via web form
- Output of temperature, humidity, condition, etc
- Basic error handling for invalid or empty input
- Light styling for clarity and responsiveness

3.4 DESIGN FLOW

- User accesses JSP page and enters city name
- Form submits request to Servlet
- Servlet makes HTTP request to weather API
- Response XML is parsed
- Parsed data is sent to JSP for rendering

3.5 DESIGN SELECTION

Using Servlets for logic and JSP for presentation follows the MVC architecture. XML was preferred for its structured format and educational value in Java web development.

3.6 IMPLEMENTATION PLAN/METHODOLOGY

- Start with HTML form inside JSP
- Write Servlet to handle input and process weather request
- Integrate XML parser to extract weather data
- Use JSP to display results using scriptlets/EL

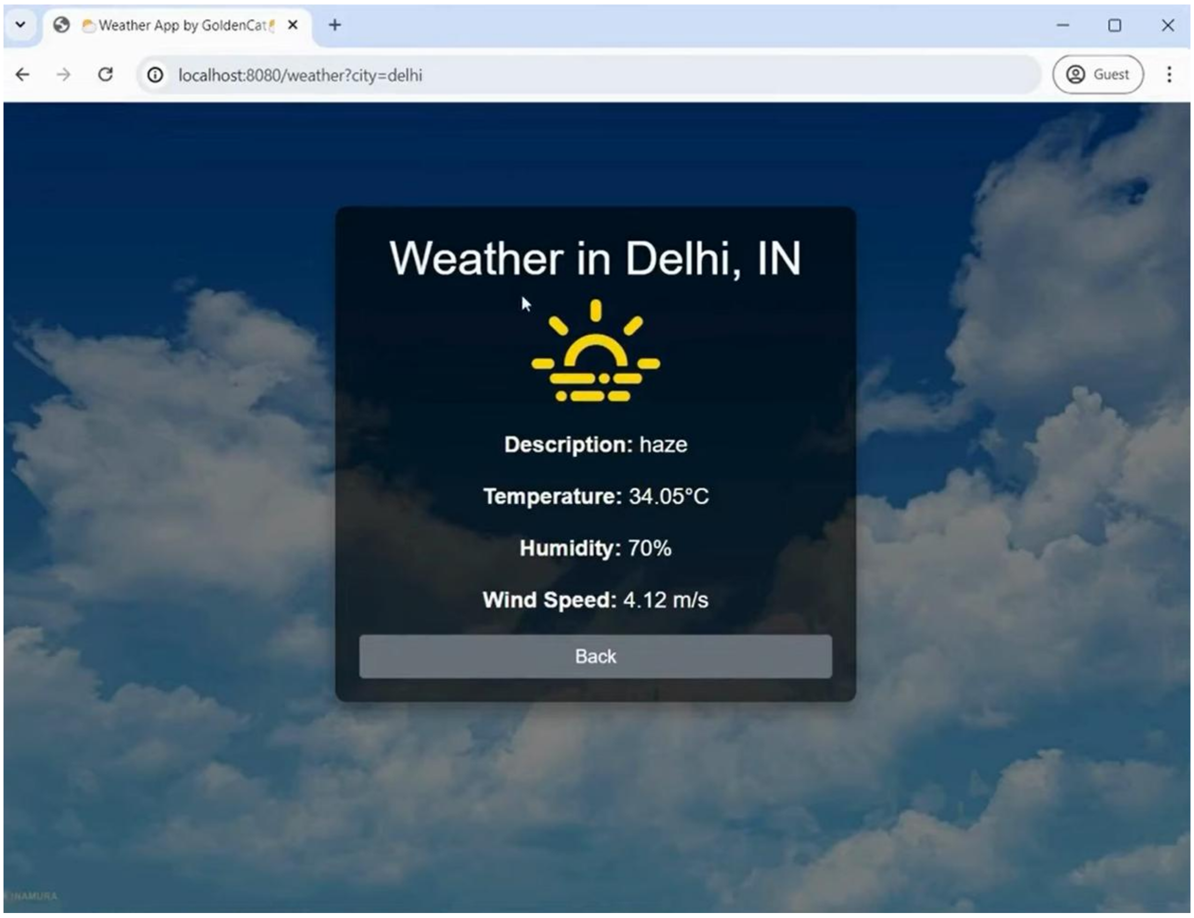
CHAPTER 4

RESULT ANALYSIS AND VALIDATION

4.1 IMPLEMENTATION OF SOLUTION

After deployment:

- The app successfully displays weather data on request
- Handles different cities globally with accurate outputs
- Error messages appear for empty or invalid input
- Consistent rendering across browsers



Weather in Delhi, IN

Description: haze

Temperature: 34.05°C

Humidity: 70%

Wind Speed: 4.12 m/s

Fig 1

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

This project developed a fully functional weather web app using Java Servlets, JSP, and XML. It demonstrates how Java can be used for practical applications and serves as an excellent case study for full-stack development in Java EE..

5.2 FUTURE WORK

1. Add forecast support (multi-day data)
2. Add graphical charts for temperature changes
3. Integrate user authentication for personalized experience
4. Use JSON format with AJAX for faster frontend updates
5. Extend to Android using Java backend APIs

REFERENCES

1. OpenWeatherMap API Documentation – https://openweathermap.org/api
2. Oracle Docs – Java Servlet and JSP Tutorials
3. W3Schools – XML Parsing in Java
4. GeeksforGeeks – Java Web App Development
5. Head First Servlets and JSP, O'Reilly Media