Building a Weather Forecast Dashboard

Group 7

- SANTHARAM S
- SARANR
- SATHURMUGAN K
- SELAPATHI V

Building a Weather Forecast Dashboard

This presentation outlines the development of a responsive weather forecasting dashboard, leveraging public APIs and modern web technologies. Designed for clarity and user engagement, it serves as a comprehensive guide for undergraduate computer science students.

CHAPTER 1

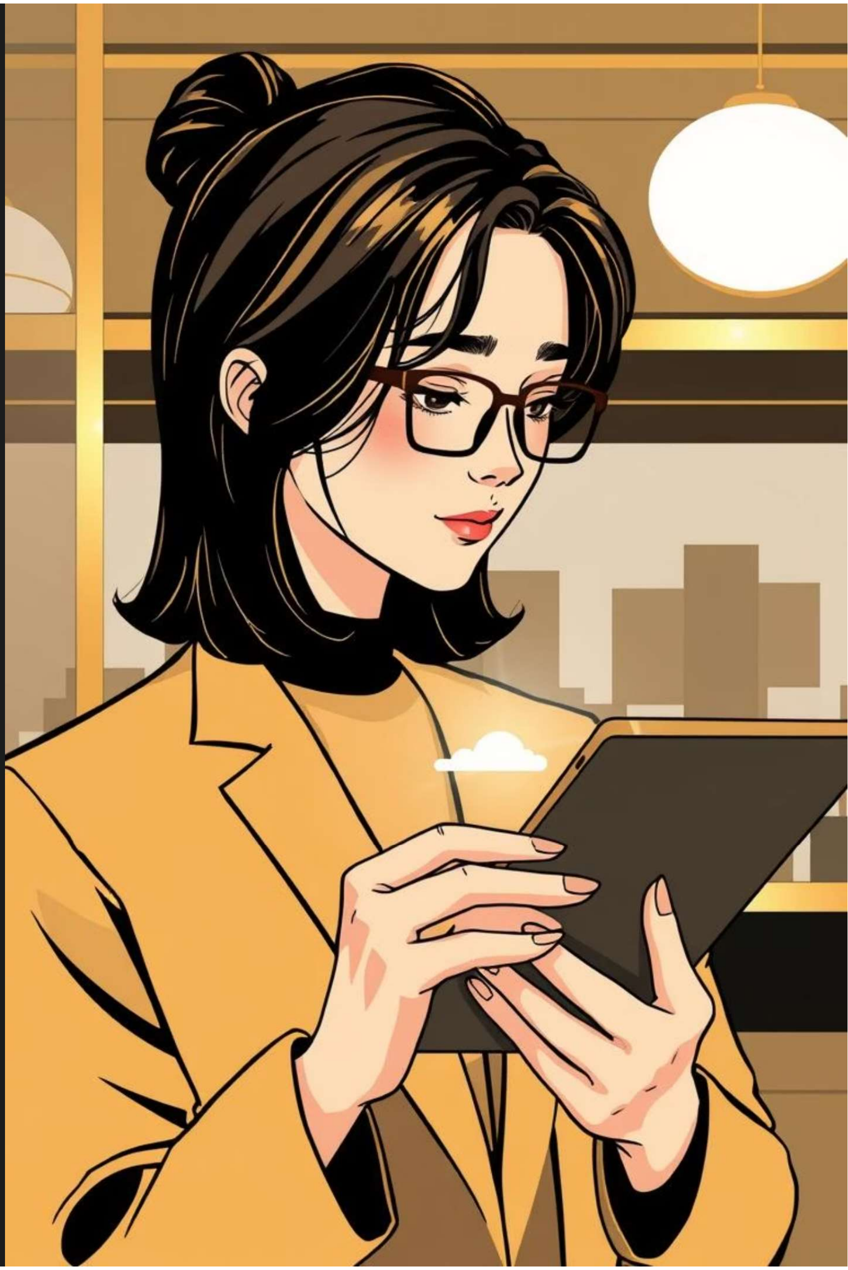
Introduction to the Project

Domain Overview

Weather forecasting is crucial for planning daily activities, travel, and even agricultural decisions. Our project delves into creating an accessible and accurate tool for this domain.

Project Purpose

The primary goal is to develop a dynamic dashboard that displays real-time weather information, demonstrating proficiency in API integration and front-end development.



CHAPTER 2

Literature Survey: Existing Systems

AccuWeather

Known for its detailed forecasts and global coverage, often incorporating radar maps and minute-by-minute updates.

The Weather Channel

Offers comprehensive weather news, videos, and personalized forecasts with an emphasis on severe weather alerts.

Google Weather

Integrated into Google Search, providing quick and concise local weather information directly in search results.

OpenWeatherMap

A popular API provider for developers, offering current weather data, forecasts, and historical data with various subscription tiers.

CHAPTER 3

Problem Statement

"Students must create a weather forecasting dashboard using a public API. They must document SDLC phases including requirement analysis, workflow design, and wireframes in Confluence/Figma. The dashboard must display temperature, humidity, and weather conditions using responsive HTML and Tailwind CSS. Extensive DOM manipulation must handle data display, search history, and dynamic updates via Fetch API. Students must manage the API integration and frontend code using GitHub with a structured commit history."

CHAPTER 4

System Requirements

Hardware Requirements

Processor	Intel Core i3 or equivalent (minimum)
RAM	8 GB (minimum)
Storage	256 GB SSD (minimum)
Display	13-inch screen with 1920x1080 resolution

Software Requirements

IDE	Visual Studio Code
Version Control	GitHub
Design Tools	Figma, Confluence
Operating System	Windows 10/11, macOS, or Linux

CHAPTER 5

SDLC Documentation

Requirement Gathering

Defining core functionalities, data points, and user interactions.

Workflow Design

Mapping data flow from API to UI, outlining user journeys.

UI Mockups (Wireframes)

Visualizing layout and component placement for responsive design.

Documentation

Using Confluence for detailed notes and Figma for design iterations.

CHAPTER 6

Implementation Details

Dashboard Features

Responsive HTML & Tailwind CSS: Ensuring optimal display across all devices.

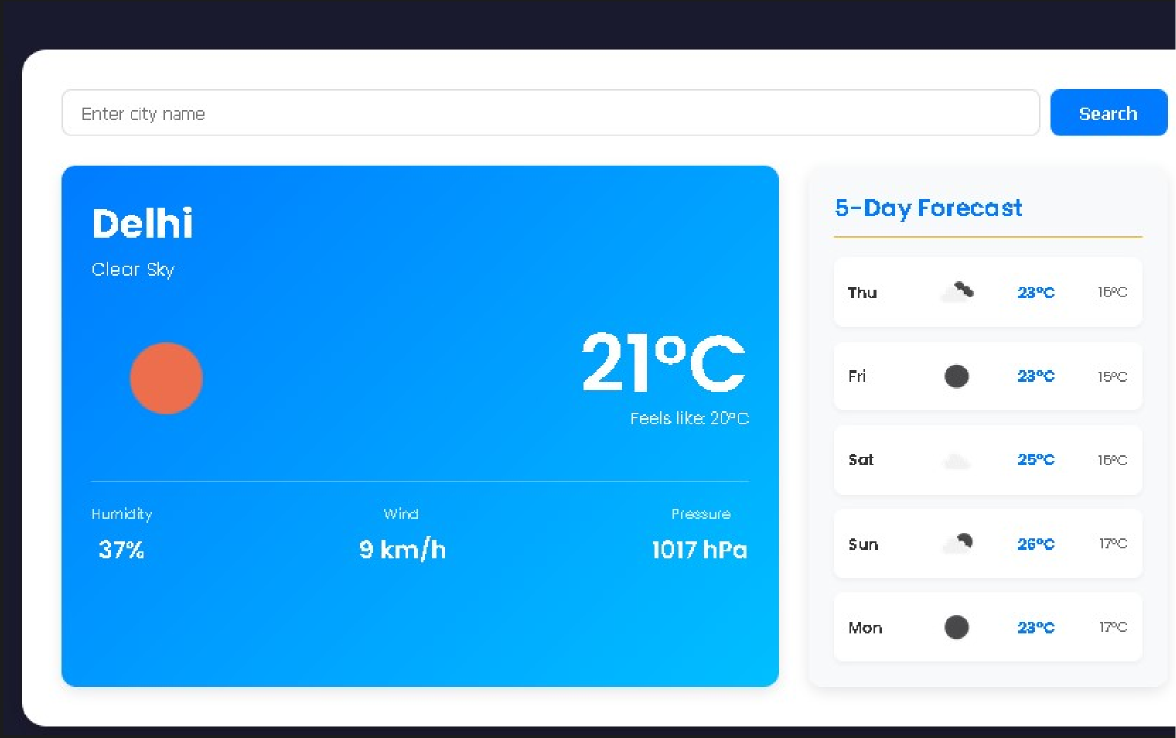
Dynamic Data Display: Temperature, humidity, and weather conditions updated in real-time.

Search History: Storing and displaying previous search queries.

Fetch API Integration: Seamlessly retrieving data from the public weather API.

DOM Manipulation: Extensive use of JavaScript for interactive elements and updates.

GitHub Management: Structured commit history for version control and collaboration.



Delhi
Clear Sky

21°C
Feels like: 20°C

Humidity	Wind	Pressure
37%	9 km/h	1017 hPa

5-Day Forecast

Thu	23°C	16°C
Fri	23°C	15°C
Sat	25°C	16°C
Sun	26°C	17°C
Mon	23°C	17°C

CHAPTER 7

Testing Procedures

Search "London"	London weather data displayed, added to history	Passed
Invalid city name	Error message displayed	Passed
Reload page with history	Search history persists	Passed
API rate limit exceeded	Graceful error handling	Passed
Resize window to mobile	Responsive layout adapts	Passed

CHAPTER 8 & 9

Results and Conclusion

Output Summary

The project successfully delivered a functional and responsive weather dashboard, showcasing key data points like temperature, humidity, and conditions.

System Performance

The dashboard demonstrated efficient data fetching and rendering, providing a smooth user experience.

Learning Outcome

Students gained hands-on experience in API integration, front-end development, and version control best practices.

CHAPTER 10 & 11

Future Enhancements & References

Future Enhancements

Extended Forecasts: Implement 5-day or 7-day weather predictions.

Location-based Services: Auto-detect user's current location.

Interactive Maps: Integrate weather radar or satellite views.

User Accounts: Allow users to save favorite locations.

Push Notifications: For severe weather alerts.

References

OpenWeatherMap API Documentation: openweathermap.org/api

Tailwind CSS Documentation: tailwindcss.com/docs

MDN Web Docs (Fetch API, DOM): developer.mozilla.org/en-US/

- "Responsive Web Design" by Ethan Marcotte