Weather Forecasting

Software Requirements Specification

Course Code:-INT222
Course Name:- Advanced Web Devlopment

Student Names: Ayush Kumar(12313610)

Prepared for
Continuous Assessment 3
Spring 2025

Table of Contents

1. Introduction

The introduction to the Software Requirement Specification (SRS) document provides an overview of the complete SRS document for the Intelligent Weather App. This document contains all the information needed by a software engineer to adequately design and implement the software product described by the requirements listed herein.

1.1Purpose

The purpose of this SRS is to describe the requirements for the "Intelligent Weather App," a web-based application designed to provide real-time weather forecasts, analytics, and intelligent activity-based advice. The intended audience for this document includes the development team, project supervisors, and potential end-users.

1.2Scope

1. Product Name: Intelligent Weather App
2. Product Description: The software will provide users with accurate weather data (current, hourly, daily) using the Open-Meteo API. It will also feature a "Smart Assistant" that analyzes weather conditions to recommend whether specific outdoor activities (running, swimming, driving) are safe or enjoyable.
3: Application:

- *Goal: To simplify weather interpretation for users by translating raw data into actionable advice.*
- *Benefits: Users save time by getting direct "Yes/No" advice for activities rather than analyzing complex weather metrics themselves.*

1.3Definitions, Acronyms, and Abbreviations

- *SRS: Software Requirements Specification*
- *API: Application Programming Interface*
- *UI: User Interface*
- *UX: User Experience*
- *MERN: MongoDB, Express.js, React, Node.js*
- *HTTP: Hypertext Transfer Protocol*
- *JSON: JavaScript Object Notation*

1.4 References

1. *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications.*
2. *Open-Meteo API Documentation: https://open-meteo.com/en/docs*

1.5 Overview

The rest of this SRS is organized as follows:

- ***Section 2** describes the general factors that affect the product and its requirements, including product perspective and user characteristics.*

- ***Section 3*** *details the specific functional and non-functional requirements.*
- ***Section 4*** *provides analysis models like Data Flow Diagrams.*
- ***Subsequent Sections*** *provide administrative details like GitHub links and client proofs.*

2. General Description

This section of the SRS should describe the general factors that affect 'the product and its requirements. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.

2.1 Product Perspective

The Intelligent Weather App is a standalone web application. It interacts with the Open-Meteo API to fetch weather data. The backend manages user authentication and session handling, while the frontend provides the interactive user interface. It replaces traditional static weather dashboards with an interactive, advisory-focused system.

2.2 Product Functions

The major functions of the software include:

- ***User Registration & Login****: Secure account creation and authentication.*
- ***Live Weather Dashboard****: Displaying current temperature, wind, humidity, and precipitation.*
- ***5-Day Forecast****: Providing a detailed outlook for the upcoming week.*
- ***Intelligent Assistant****: An AI-like chatbot interface that answers questions like "Can I go for a run?" based on pre-defined weather rules.*
- ***Analytics****: Visualizing weather trends (temperature, humidity) over time using graphs.*

2.3 User Characteristics

- ***General Users****: Individuals looking for daily weather updates.*
- ***Outdoor Enthusiasts****: Runners, swimmers, and hikers who need specific weather conditions.*
- ***Travelers****: Users checking weather for different cities.*
- ***Technical expertise****: Users are expected to have basic web browsing skills*

2.4 General Constraints

- ***Internet Connection****: The application requires an active internet connection to fetch real-time data.*
- ***API Limits****: The Open-Meteo API usage relies on their free tier limits (though generally generous).*
- ***Browser Compatibility****: Optimised for modern web browsers (Chrome, Firefox, Edge).*

2.5 Assumptions and Dependencies

- ***Assumption***: *The Open-Meteo API will remain available and free to use.*
- ***Assumption***: *Users will provide valid city names for weather lookups.*
- ***Dependency***: *The application depends on Node.js and MongoDB being installed/available for the backend.*

3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be:

- *Correct*
- *Traceable (both forward and backward to prior/future artifacts)*
- *Unambiguous*
- *Verifiable (i.e., testable)*
- *Prioritized (with respect to importance and/or stability)*
- *Complete*
- *Consistent*
- *Uniquely identifiable (usually via numbering like 3.4.5.6)*

Attention should be paid to the carefuly organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

3.1 External Interface Requirements

3.1.1 User Interfaces

- ***Navbar***: *Persistent navigation for Home, Forecast, Analytics, and Assistant.*
- ***Dashboard***: *Clean, card-based layout for weather metrics.*
- ***Assistant Chat***: *A chat-like interface for querying the smart advisor.*
- ***Responsive Design***: *The UI shall adapt to desktop, tablet, and mobile screens.*

3.1.2 Hardware Interfaces

- ***Server***: *Standard server hardware capable of running Node.js.*
- ***Client***: *Any device with a modern web browser and internet connectivity.*

3.1.3 Software Interfaces

- ***Database***: *MongoDB for storing user credentials.*

- ***Operating System**: Cross-platform (Windows, Linux, macOS).*
- ***External API**: Open-Meteo API for weather data.*

3.1.4 Communications Interfaces

- *All data transfer between client and server shall occur over HTTP/HTTPS.*
- *The server listens on a configurable port (default 5000).*

3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.2.1 User Authentication

- ***3.2.1.1 Introduction**: Users must be able to create accounts and log in to save preferences (future scope).*
- ***3.2.1.2 Inputs**: Email, Password, Full Name.*
- ***3.2.1.3 Processing**: Passwords are hashed using bcrypt before storage. Unique email check is performed.*
- ***3.2.1.4 Outputs**: Success message and redirection to dashboard upon login.*
- ***3.2.1.5 Error Handling**: Invalid credentials return a 401 Unauthorized error.*

3.2.2 Weather Data Retrieval

- ***3.2.2.1 Introduction**: The system fetches weather data for a specified city.*
- ***3.2.2.2 Inputs**: City name (via search bar or passed to API).*
- ***3.2.2.3 Processing**: The backend/frontend queries the Open-Meteo API with the city's coordinates.*
- ***3.2.2.4 Outputs**: JSON object containing temperature, wind speed, humidity, etc.*
- ***3.2.2.5 Error Handling**: If a city is not found, an appropriate error message is displayed.*

3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

3.5.1 Performance

- *Weather data should load within 2 seconds on a standard 4G connection.*
- *The application should support concurrent users without significant lag.*

3.5.2 Reliability

- *The system handles API failures gracefully by showing cached data or a user-friendly error.*

3.5.3 Availability

- *The application should be available 99% of the time during business hours.*

3.5.4 Security

- *User passwords must always be hashed.*
- *API endpoints should validate user sessions where necessary.*

3.5.5 Maintainability

- *The code is modular (Frontend separate from Backend).*
- *React components are reusable.*

3.5.6 Portability

- *The web app is accessible from any device with a browser.*

3.7 Design Constraints

- *Must use the MERN stack (MongoDB, Express, React, Node).*
- *Must use functional components and Hooks in React.*

3.9 Other Requirements

- *The UI must be aesthetically pleasing with modern color schemes (e.g., glassmorphism).*

4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.

4.1 Data Flow Diagrams (DFD)

A. Appendices

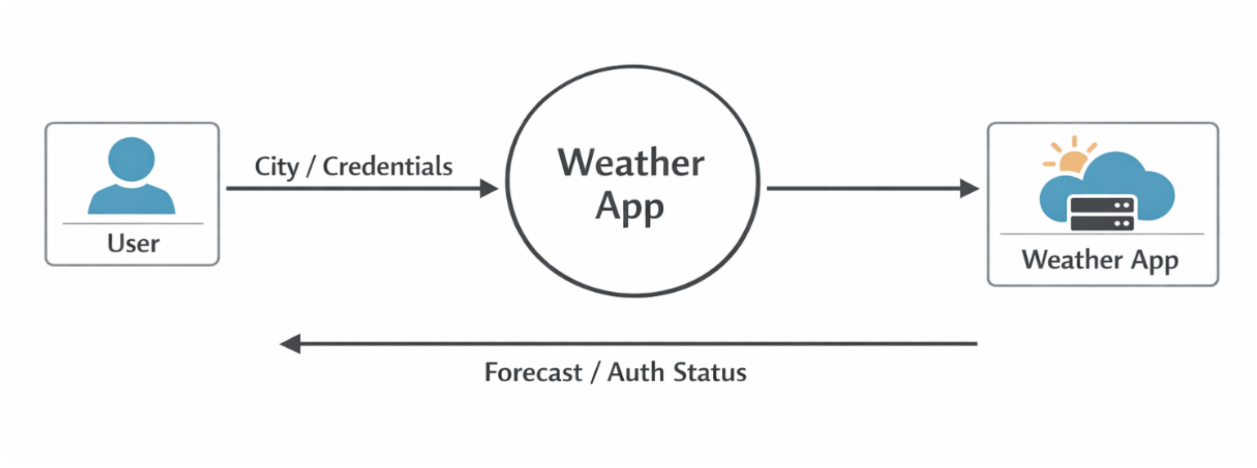
Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements.

Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.

Level 0 DFD*:*

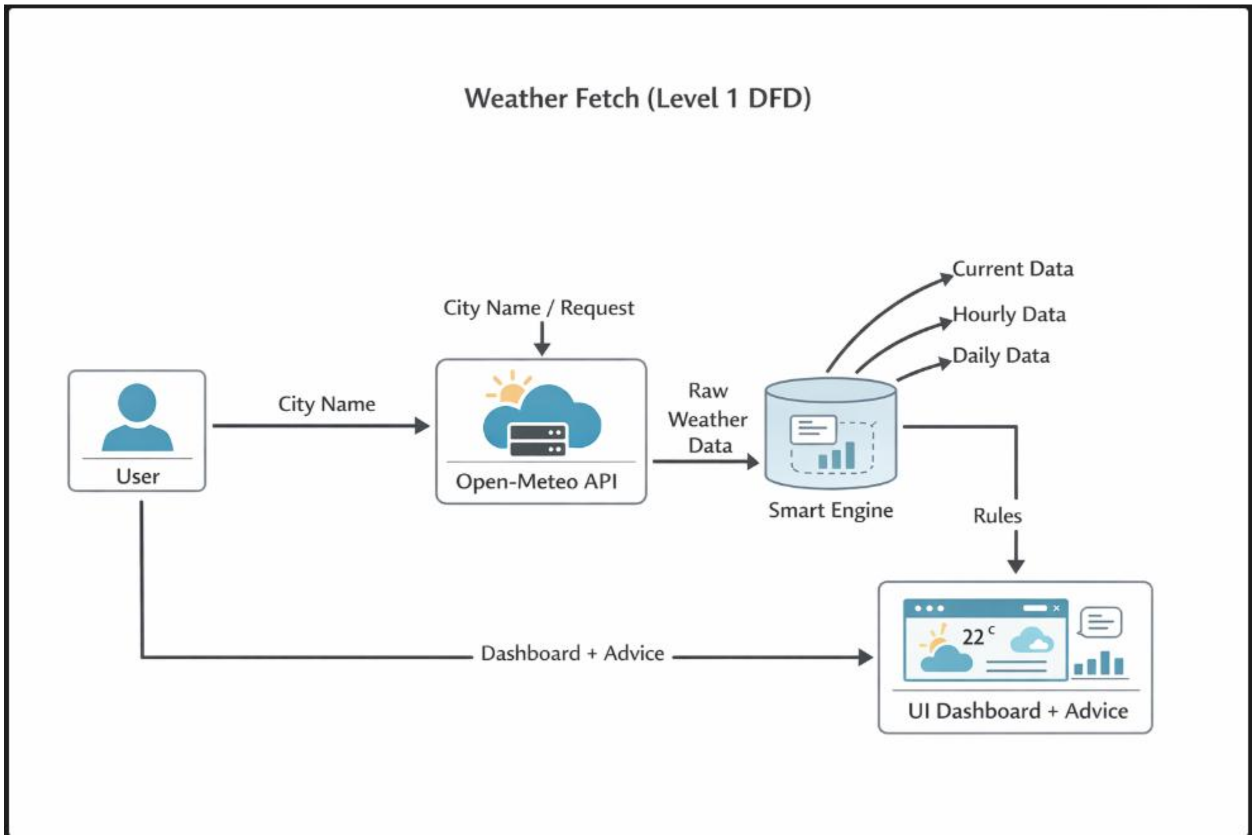
- ***User*** *-> sends* ***City/Credentials*** *->* ***Weather App***

- ***Weather App** -> returns **Forecast/Auth Status** -> **User***



Level 1 DFD (Weather Fetch):

1. ***User** inputs **City Name**.*
2. ***Frontend** sends request to **Open-Meteo API** (or via proxy).*
3. ***API** returns **Raw Weather Data**.*
4. ***Frontend** processes data into **Current/Hourly/Daily** objects.*
5. ***Smart Engine** checks rules against **Weather Data**.*
6. ***UI** displays **Dashboard** + **Advice**.*



Deployed Link-https://weatherapp-frontend-dg8z.onrender.com/

Github: https://github.com/ayushkumar87/weather-app
Social media: https://youtu.be/D0b9igtejiw

Class Link-
https://github.com/ayushkumar87/weather-app/tree/main/INT252