SOFTWARE REQUIREMENTS SPECIFICATION

**For**

**Weather App with DB**

**Prepared by:-**

*Arpita Viswanathan*

*Dharshini R*

*Gayathri V*

# Introduction

## Purpose

## The main objective of this document is to illustrate the requirements of the project Weather App with DB. Our application not only provides real-time weather updates, but it also incorporates a database for the storage and retrieval of historical weather information. A comprehensive log that displays the history of weather updates is easily accessible to users from within the app's source document. Users may easily monitor changes over time with this functionality. You may make well-informed judgments based on historical data by using our app, which gives you constant access to the most recent weather information as well as a thorough history of previous conditions.

## Document Conventions

* + - Entire document should be justified.
    - Convention for Main title

Font face: Times New Roman Font style: Bold

Font Size: 14

* + - Convention for Sub title

Font face: Times New Roman Font style: Bold

Font Size: 12

* + - Convention for body

Font face: Times New Roman Font Size: 12

## Scope of Development Project

## Real-time weather integration and historical data management are two essential components that are included in the development scope of our weather app. First and foremost, we want to build an intuitive user interface that shows the current weather conditions without any problems, using data from dependable APIs or services. In an easy-to-use layout, this interface will display important meteorological parameters like temperature, humidity, wind speed, and more. In order to store and retrieve historical weather data, we'll concurrently develop a strong backend system that integrates with a database. Users will be able to easily examine and compare past weather updates thanks to a log that is conveniently located within the app's source document.

## The development process will prioritize effective data management techniques to guarantee that the database efficiently arranges past records for prompt retrieval. We'll also give security measures top priority in order to protect user data and uphold the accuracy of historical and real-time weather data. Thorough testing and optimization will be carried out during the development cycle to guarantee the app functions properly on a variety of devices and in a range of network situations. Our app seeks to provide users with meaningful weather information so they may make informed decisions by fusing real-time updates with an extensive historical log.

## Definitions, Acronyms and Abbreviations

JAVA -> platform independence SQL-> Structured query Language ER-> Entity Relationship

UML -> Unified Modeling Language

IDE-> Integrated Development Environment SRS-> Software Requirement Specification

## References

* + - Books

 Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices (ACM Press) by Michael Jackson

Software Requirements (Microsoft) Second EditionBy Karl E. Wiegers

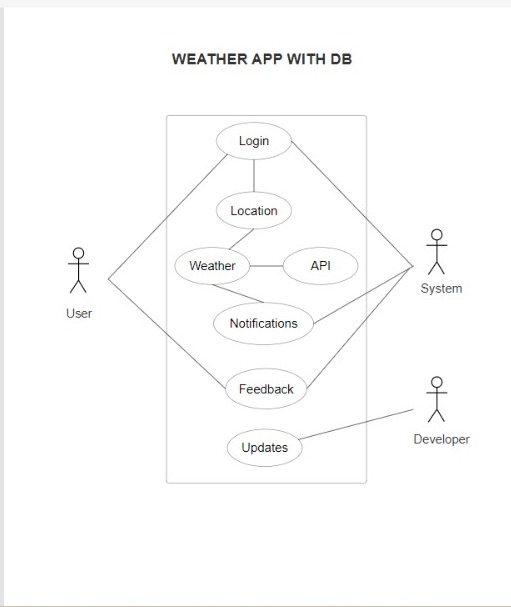
Software Engineering: A Practitioner’s Approach Fifth Edition By Roger S. Pressman

* + - Websites
* [**http://www.slideshare.net/**](http://www.slideshare.net/)
* [**https://www.scribd.com/document/534112346/Weather-App-Using-Android-Studio**](https://www.scribd.com/document/534112346/Weather-App-Using-Android-Studio)

# Overall Descriptions

## Product Perspective

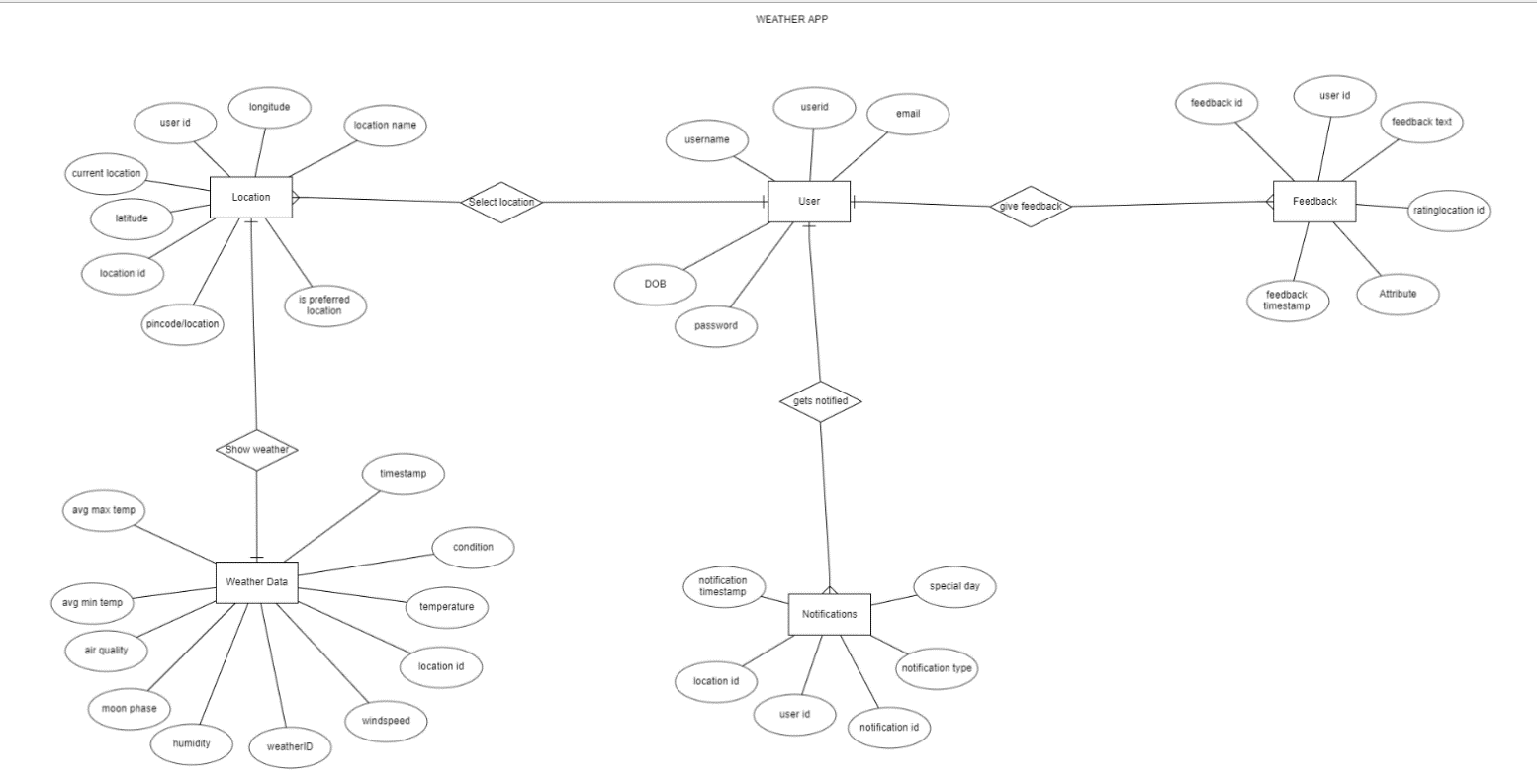
Use Case Diagram of Weather App with DB.



Real-time weather updates and forecasts are provided by the Weather App through database integration. In order to guarantee a safe and dependable experience, users can control locations and interact with features like book selection and notifications. An easy-to-use UI and comprehensive documentation add to the app's viewpoint.

## Product Function

Entity Relationship Diagram of Weather App with DB.



Users may establish accounts, manage locations, and get forecasts and updates on the weather in real time using the Weather App with Database. Notifications, a feedback system, and book selection are further elements. The application places a high priority on performance, security, and regulatory compliance while maintaining a user-friendly interface. Extensive documentation enhances the usability of the program and makes it expandable for further developments.

## User Classes and Characteristics:

The system provides different types of services based on the type of users . The User will be able to access the weather of different locations. The controllers will be able to access the database and change it if necessary.

The features that are available to the controller are:-

* + - Establish and enforce robust security protocols to safeguard user data and ensure the integrity of weather information.
    - Provide tools to customize and manage user preferences, such as notification settings, default locations, and display preferences.
    - Establish and maintain partnerships with meteorological sources to enhance the accuracy and reliability of weather data.
    - Implement efficient data storage strategies and optimize database queries to enhance the speed and responsiveness of the app.
    - Perform routine audits of user accounts, system logs, and database records to identify and address any irregularities or discrepancies.
    - Develop and execute robust backup and recovery procedures to ensure data integrity and minimize the impact of potential system failures.
    - Keep abreast of technological advancements in meteorology and database management to continuously improve the app's features and capabilities.
    - Work closely with the development team to implement new features, address system vulnerabilities, and enhance overall system performance.
    - Develop training materials and conduct sessions to educate users, particularly advanced users, on utilizing advanced features and interpreting detailed weather reports.
    - Generate and analyze periodic reports summarizing user engagement, system performance, and any noteworthy trends or patterns.

The features that are available to the Members are:-

* + - Check out the local weather conditions right now.
    - Look for weather predictions for particular dates.
    - Decide on your preferred destinations and set preferences.
    - Obtain customized weather alerts.
    - Comment on how user-friendly the app is.
    - Obtain historical meteorological data.

## Operating Environment

## Our weather app runs on a cross-platform environment, which means it works with a range of gadgets and operating systems, including web browsers, iOS, and Android. In order to retrieve weather data in real time, it will make use of APIs or services, keeping in touch with dependable sources to obtain up-to-date meteorological information. For real-time updates and data synchronization with the database—which is housed on secure servers to guarantee user privacy and data integrity—the app will need to be connected to the internet. Furthermore, the interface will be made to work and navigate fluidly on a variety of screen sizes, offering a unified user experience on any platform or device.

## Assumptions and Dependencies

The assumptions are:-

* + - The coding should be error free
    - The system should be user-friendly so that it is easy to use for the users
    - The system should have more storage capacity and provide fast access to the database
    - The system should provide search facility and support quick transactions
    - Users may access from any computer that has Internet browsing capabilities and an

Internet connection

* + - Users must have their correct usernames and passwords to enter into their online accounts and do actions
    - Users should be able to read the now and past weather information at ease.

The dependencies are:-

* + - The specific hardware and software due to which the product will be run
    - On the basis of listing requirements and specification the project will be developed and run
    - The end users should have proper understanding of the product
    - The system should have the general report stored

## Requirement

Software Configuration:-

This software package is developed using java as front end which is supported by sun micro system. Microsoft SQL Server as the back end to store the database.

Operating System: Windows NT, windows 98, Windows XP Language: Java Runtime Environment, Net beans 7.0.1 (front end) Database: MS SQL Server (back end)

Hardware Configuration:- Processor: Pentium(R)Dual-core CPU Hard Disk: 40GB

RAM: 256 MB or more

## Data Requirement

## With an emphasis on user interactions and account features, the weather app project gathers vital information such as email addresses, passwords, and usernames. By configuring default and saved locations, users customize their experience. Users can obtain timestamps and account creation dates from the query outputs, which also include location settings, weather requests, and account creation. The database schema optimizes performance for effective data retrieval by including tables for users, locations, weather information, and preferences. The system is tested, complies with regulations, follows security protocols, and has extensive documentation. A dependable and scalable program is enhanced by regular data backups that guarantee integrity.

# External Interface Requirement

## GUI

**Weather API Interface:**

Fetches current weather data for a specific location,weather forecast data for a defined future period and historical weather data for a specified location and time range.

**Database Interface: Tables present : Location Table:** Stores information about different locations for which weather data is being tracked with the fields of Location ID, City Name, Latitude, Longitude, Country, etc. **Current Weather Table:** Stores current weather data for various locations with the fields ofWeather ID, Location ID (foreign key), Temperature, Humidity, Wind Speed, Weather Condition, Timestamp, etc. **Forecast Table:** Stores forecasted weather data for different time periods and locations with the fields of Forecast ID, Location ID (foreign key), Date, Forecasted Temperature, Forecasted Weather Condition, Forecasted Humidity, etc.

**DBMS Selection:** 1. Scalability 2. Data Complexity 3.Requirements

**Connection Parameters:** 1.Hostname 2.Port 3.Database Name 4.Username and Password

**User** **Interface (UI):** Designing the user interface (UI) for a weather app involves creating a visually appealing layout that provides easy access to weather information, enables location search, incorporates user settings, and ensures usability and accessibility for all users.

**Design & Layout:**

Ensure a clean and intuitive design that guides users to easily find and understand weather information without unnecessary complexity.

Use clear labels, icons, and visual cues to assist users in navigating the app. 1.Weather Display Components 2.Location Search 3.User Settings 4.Navigation 5.Usability & Accessibility.

**Security and Privacy:**

**Encryption and Hashing:**

Use encryption techniques (e.g., AES, RSA) to secure sensitive data stored in the database, such as user credentials, API keys, and other critical information. Ensure that the encryption keys are kept secure.

Implement hashing algorithms (e.g., bcrypt, SHA-256) to securely store passwords. Store only the hashed version of passwords in the database to prevent unauthorized access even if the data is compromised.

**API Key Protection:**

Store API keys and other sensitive information in secure environments or configuration files that are not accessible to unauthorized users.

Utilize environment variables or secret management tools to securely manage and access API keys within your application code.

**HTTPS Protocol:**

Ensure that all communications between the weather app, API, and database occur over HTTPS (Hypertext Transfer Protocol Secure). HTTPS encrypts the data transmitted between the client (app) and server (API or database), safeguarding it from interception or tampering.

Configure SSL/TLS certificates for the server hosting your application and API to enable secure HTTPS connections.

**SSL/TLS Implementation:**

Enforce SSL/TLS protocols for secure connections between the app and external services (API, database). Use modern and secure TLS versions and cipher suites.

Regularly update SSL/TLS configurations and certificates to mitigate potential security vulnerabilities.

**Data Transmission Security:**

Implement secure authentication and authorization mechanisms for access to the API and database. Use tokens or API keys in the authorization headers of requests.

Employ additional security measures such as API rate limiting, access control lists (ACLs), and firewalls to prevent unauthorized access and mitigate potential attacks.

**Data Processing and Storage:**  **API Response Parsing:** Retrieve weather data from the API endpoints and Parse the received API responses, typically in JSON or XML format, using programming language libraries or built-in functions to extract relevant weather information fields.

**Data Transformation:** Transform the extracted data into a format suitable for storage in the database. **Normalization and Cleaning:** Normalize the data by organizing it into appropriate tables and fields within the database to Perform data cleaning tasks by avoid redundancy and maintain consistency. **Validation and Verification:** Implement validation checks to ensure that the received data meets specified criteria like completeness of data by running sanity checks before insertion into the database. **Constraints and Foreign Keys:** Use foreign keys to establish relationships between tables and preventing orphaned data.

**Error Handling:** set up error handling mechanism to identify and handle Log errors or exceptions encountered during data processing for later analysis and debugging purposes.

# System Features

# Users of the weather app can manage locations, register, and get predictions and current weather information. It keeps a user-friendly design for easy navigation while ensuring data security with authentication and encryption. The performance and dependability of the system are enhanced by rigorous testing, documentation, and backup plans.Other Non-functional Requirements

## Performance Requirement

Quick access to historical weather data and fast real-time updates are the main performance requirements for our weather app. Ensuring that meteorological information loads quickly and that retrieving previous data is effective even for large amounts of data, it strives for a flawless user experience. The application is designed to be as responsive as possible, so users may quickly switch between areas and get historical and current weather data.

**Real-time Updates:** Ensure weather data loads within 2-3 seconds upon app launch.

**Efficient Historical Data Retrieval:** Access to past weather updates should take no more than 3-4 seconds.

**Responsive Navigation:** Smooth transitions between app sections, ensuring minimal lag or delay.

**Optimized Database Operations:** Quick and seamless data fetching and storage within the app, even with a large volume of historical records.

**Consistent Performance Across Platforms:** Maintain similar performance standards across various devices (iOS, Android) and web browsers.

## Safety Requirement

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup so that the database is not lost. Proper UPS/inverter facility should be there in case of power supply failure.

## Security Requirement

* + - System will use secured database
    - Normal users can just read information but they cannot edit or modify anything except their personal and some other information.
    - System will have different types of users and every user has access constraints
    - Proper user authentication should be provided
    - No one should be able to hack users’ password
    - There should be separate accounts for admin and members such that no member can access the database and only admin has the rights to update the database.

## Requirement attributes

**Performance:** To provide quick user interaction, the app should load historical weather data in 3–4 seconds and current weather data in 2-3 seconds.

**Reliability:** In order to access historical data retrieval and real-time weather updates, the system must maintain at least 99% uptime.

**Scalability:** As user data expands, the application's database should be able to accommodate more requests for data storage and retrieval without sacrificing user experience.

**Security:** To protect user data and maintain confidentiality and integrity throughout data transfer and storage, implement encryption techniques.

**Compatibility:** Make sure the app is compatible so that users can use it on a variety of platforms and devices (iOS, Android, and the web) and have a consistent experience.

**Usability:** Preserve a user interface that is simple to navigate and accessible to people with different levels of technological proficiency.

**Maintainability:** Develop the app with modular components for easy updates and maintenance, ensuring scalability and future enhancements.

## Business Rules

A business rule is anything that captures and implements business policies and practices. A rule can enforce business policy, make a decision, or infer new data from existing data. This includes the rules and regulations that the System users should abide by. This includes the cost of the project and the discount offers provided. The users should avoid illegal rules and protocols. Neither admin nor member should cross the rules and regulations.

## User Requirement

Our weather app must provide users with instant access to current weather conditions based on their location and an easy-to-use interface that shows important characteristics like wind speed, temperature, and humidity. The ability to easily retrieve previous weather updates, compare data across multiple timeframes, and maybe add more elements like graphs or charts for improved data visualization are among the things that users want to be able to do with ease while exploring historical weather data. Users also want a dependable, safe software that protects their privacy and offers access to precise, comprehensive weather data for both casual and in-depth study, taking into account a variety of user skill levels and preferences.

# Other Requirements

## Data and Category Requirement

Real-time meteorological data, including temperature, humidity, wind speed, precipitation totals, and atmospheric pressure, from dependable APIs or services, is needed for our weather app. These data categories have ideally contain historical records that go back at least a few months or years, giving consumers access to previous weather reports for certain places and periods of time. Users should be able to search and retrieve historical data by date, location, or weather parameter by organizing it chronologically. To guarantee precise referencing and quick data retrieval for a range of analytical uses inside the app, the database should also save metadata, such as timestamps and position IDs.

## Appendix

A: Abbreviation, Acronym, Assumptions; B: Business rules; C: Class, Client, Conventions; D: Data requirement, Dependencies; G: GUI; K: Key; M: Member; N: Non-functional Requirement; O: Operating environment; P: Performance, Perspective, Purpose; R: Requirement, Requirement attributes; S: Safety, Scope, Security, System features; U: User, User class and characteristics, User requirement;

## Glossary

The following are the list of conventions and acronyms used in this document and the project as well:

* + - User: A general login id assigned to most users
    - Client: Intended users for the software
    - SQL: Structured Query Language; used to retrieve information from a database
    - SQL Server: A server used to store data in an organized format
    - Layer: Represents a section of the project
    - User Interface Layer: The section of the assignment referring to what the user interacts with directly
    - Application Logic Layer: The section of the assignment referring to the Web Server. This is where all computations are completed
    - Data Storage Layer: The section of the assignment referring to where all data is recorded
    - Use Case: A broad level diagram of the project showing a basic overview
    - Class diagram: It is a type of static structure diagram that describes the structure of a system by showing the system’s cases, their attributes, and the relationships between the classes
    - Interface: Something used to communicate across different mediums
    - Unique Key: Used to differentiate entries in a database

## Class Diagram

Key classes and their relationships are shown in the class diagram for the Weather App with Database SRS. The "User" class reflects user interactions using methods like SetPreferences and characteristics like UserID. The "Location" class links to the "WeatherData" class, which records meteorological data, and represents geographic locations with features like latitude and longitude. The "Notification" and "Feedback" classes control how users interact with notifications and submit feedback. System control is supervised by the "Administrator" class, and database-related tasks are performed by the "DatabaseManager" class. Location-specific tasks fall under the purview of the "LocationManager" class. One-to-many links between users and the destinations they have chosen, users and feedback, and users and notifications are examples of relationships. The structure and interactions within the Weather App with Database are shown in a static form in the diagram.

