# WeatherPlus: Your Personalized Weather Companion

Keyur Patel - 16010421073
Ronit Mehta - 16010421056
Batch - A2
Exp-2(SRS)

February 6, 2024

## Aim:

Preparation of Software Requirements Specifications (SRS).

## Resources:

LaTeX Editor, Internet Browser

# 1 Introduction

### 1.1 Product Overview:

- **Purpose:** The primary objective of WeatherPlus is to offer a feature-rich weather experience, allowing users to access up-to-the-minute weather updates, detailed forecasts, and personalized settings. Whether planning a day out or staying aware of the changing climate, WeatherPlus aims to provide a seamless and intuitive platform for users to make informed decisions based on the latest meteorological data.
- **Environment:** WeatherPlus operates in a digital environment, compatible with various platforms, including smartphones, tablets, and desktops. The application leverages advanced meteorological data sources to ensure the accuracy of information, making it a reliable companion for users in diverse geographical locations.
- **Target Users:** WeatherPlus is designed for users of all levels, irrespective of their educational background or technical expertise. The application's intuitive interface caters to individuals with varying levels of experience in using weather applications, making it accessible to both tech-savvy users and those who may be less familiar with such tools.

# 2 Specific Requirements

### 2.1 External Interface Requirements:

#### 2.1.1 User Interface:

1. ***Real-Time Weather Updates Display:*** The interface should prominently display real-time weather updates such as temperature, humidity, wind speed, and weather descriptions.

2. ***Location-Based Services Integration:*** The app should have a feature that automatically detects the user's location using geolocation services.

3. ***Current Conditions at a Glance:*** The main dashboard should provide a snapshot of the current weather conditions.

4. ***Comprehensive Forecasts:*** The app should offer detailed weather forecasts, including hourly and daily predictions.

5. ***Personalized User Experience:*** WeatherPlus should allow users to personalize their experience by saving favorite locations for quick access.

6. ***Intuitive User Interface Design:*** The interface should be intuitive and user-friendly, prioritizing clarity and ease of use. This means employing clear navigation elements, intuitive icons, and easily understandable visual cues to help users find the information they need quickly and efficiently.

7. ***Accessibility Considerations:*** The app should be designed with accessibility in mind, ensuring that users with disabilities can navigate and interact with the interface effectively.

#### 2.1.2 Software Interfaces:

1. **Required Software Products:**

   - ***Database System:***
     Name and Version: PostgreSQL 13.4
   - ***Operating System:***
     Name and Version: Windows 11 23H2
   - ***APIs for Weather Data:***
     Name and Version: OpenWeatherMap API, Version 2.5

2. **Interfaces:**

   - ***Database Interface:***
     Purpose: The database system stores user preferences, historical weather data, and other relevant information.
   - ***Operating System Interface:***
     Purpose: The website relies on the operating system for file management and server-side operations.
   - ***Weather API Interface:***
     Purpose: Fetches real-time weather data to display on the website.

#### 2.1.3 Communications Protocols:

1. **HTTP/HTTPS for OpenWeatherMap API Communication:**

   - WeatherPlus communicates with the OpenWeatherMap API over HTTP/HTTPS to fetch real-time weather updates and forecasts. The use of HTTPS ensures secure transmission of sensitive data.
   - WeatherPlus sends HTTP requests to specific endpoints provided by the OpenWeatherMap API. These requests include parameters such as location coordinates, and the API responds with JSON-formatted weather data.

2. **Platform-Specific APIs (Core Location for website):**

   - WeatherPlus utilizes platform-specific APIs for geolocation services to determine the user's current location automatically.
   - These interfaces involve making calls to specific functions within the APIs to retrieve latitude and longitude information.

## 2.2 Software Product Features:

1. **Functional Requirements**

   - **Real-Time Weather Updates**
     (a) The system shall perform validity checks on location inputs to ensure they are within acceptable geographical coordinates.
     (b) The system shall execute a sequence of operations to retrieve real-time weather data from the OpenWeatherMap API based on the user's location.
   - **Location-Based Services**
     (a) The system shall validate user permissions for geolocation services to ensure proper functioning.
     (b) The system shall implement error handling and recovery for situations where manual input of location is required, ensuring a user-friendly experience.
   - **Current Conditions at a Glance**
     (a) The system shall validate the user-selected location for displaying current weather conditions.
     (b) The system shall sequence operations to retrieve and display the current weather conditions for the selected location.
   - **Personalized User-Experience:**
     (a) The system shall validate user preferences for temperature units and language selections.
     (b) The system shall sequence operations to apply and save user preferences for future sessions.
   - **Sunrise and Sunset Times:**
     (a) The system shall display information about sunrise and sunset times for the user's current location or selected locations.
   - **Air Quality Index:**
     (a) The system shall include air quality information, such as the AQI, for the specified location.

## 2.3 Software System Attributes:

### 2.3.1 Reliability:

- A reliable weather app should provide accurate and up-to-date information. Regular testing and updates are crucial to ensuring reliability.
- The application should strive for a high level of service uptime, such as achieving 99.9% availability.

### 2.3.2 Availability:

1. ***Check Pointing:***
   - Implement regular check-pointing at predefined intervals, considering factors such as the frequency of data updates and user interactions.
   - Save critical data, including user preferences, favorite locations, current weather conditions, and forecast data during each check point.
2. ***Recovery:***
   - Have a plan to backup all important data, so even if something goes wrong, you can recover it.
   - Consider using incremental backups to reduce the time and resources required for regular backups.

3. ***Restart:***
   - When you need to stop the app, do it in a safe way to avoid causing problems.

#### 2.3.3 Security:

- Implement robust user authentication mechanisms to verify the identity of users. This includes secure login methods like multi-factor authentication to enhance account security.
- If the app integrates with external APIs for weather data, ensure that the connections are secure. Validate and sanitize data received from external sources to prevent security vulnerabilities like injection attacks.

#### 2.3.4 Maintainability:

1. ***Modularity :*** Design the application with modular components that are independent and can be updated or replaced without affecting the entire system.
2. ***Documentation:*** Good documentation facilitates easier understanding of the codebase, accelerates onboarding of new developers, and aids in troubleshooting and updates.
3. ***Version Control:*** Utilize a version control system (e.g., Git) and ensure that all code changes are appropriately versioned.
4. ***Reusability of Code:*** Encourage the reuse of code where applicable, promoting the creation of libraries or modules that can be used across different parts of the application.

#### 2.3.5 Portability:

1. ***Portable Language Usage:*** Develop the core functionalities of the weather app using a proven portable language, such as Python, html,css and JavaScript.
2. ***Responsive Design:*** Implement a responsive design for web-based components, ensuring optimal display and functionality across a range of devices and screen sizes.

#### 2.3.6 Performance:

1. ***Static Numerical Requirements:***
   - Support a minimum of 10 simultaneous users. Ensures the app can handle a significant user load without degradation in performance during normal usage.
2. ***Dynamic Numerical Requirements:***
   - Update real-time weather data at least every 15 minutes. Ensure that users receive timely and accurate information, especially during rapidly changing weather conditions.

### 2.4 Database Requirements:

1. ***Types of Information:***

- *User Information:*
  - UserID (Primary Key)
  - Username
  - Email
  - Password
- *Location Information:*
  - LocationID (Primary Key)
  - City
  - Latitude
  - Longitude
- *Weather Data:*
  - DataID (Primary Key)
  - Temperature

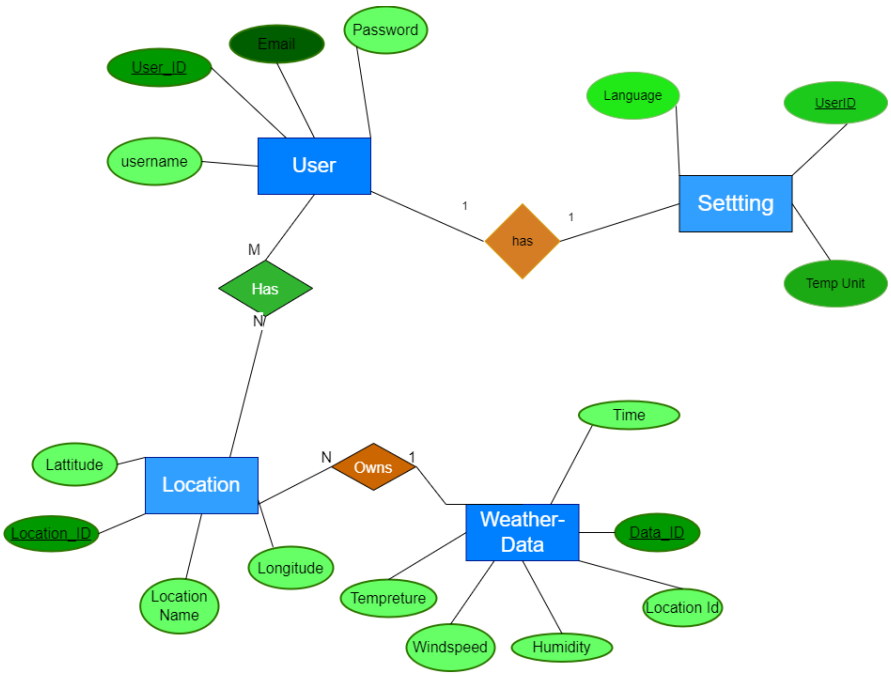


Figure 1: Weather-app ER diagram

  - Humidity
  - WindSpeed
  - Date
  - Time
- *Settings:*
  - UserID (Primary Key)
  - Temperature Unit
  - Language

2. ***Data Entities and Their Relationships:***
   - *User-Location Relationship:*
     - Many-to-Many relationship between User and Location.
     - Each user can have multiple saved locations.
   - *Location-WeatherData Relationship:*
     - One-to-Many relationship between Location and WeatherData.
     - Each location can have multiple sets of weather data.

# Questions:

**Explain various steps involved in Requirement Engineering.**

**Ans:**

- **Feasibility Study:** Assess the viability of the proposed system.
- **Requirement Elicitation:**Gather information from stakeholders through interviews, surveys, and workshops.
- **Requirement Analysis:** Organize, categorize, and analyze gathered requirements for clarity and consistency.

- **Requirement Specification:** Document requirements clearly using natural language, diagrams, and tables.
- **Requirement Validation:** Verify and ensure requirements are complete, consistent, and feasible.
- **Requirement Management:** Establish a process for handling changes to requirements and maintain version control.
- **System Modeling and Prototyping:** Develop visual models or prototypes to validate and refine requirements.
- **Requirements Traceability:** Establish links between requirements and other project artifacts.
- **Communication:** Ensure effective communication between stakeholders and the development team.
- **Documentation and Maintenance:** Create and maintain comprehensive documentation for requirements.

# Outcomes

**CO3 :** Demonstrate requirements, modeling and design of a system

# Conclusion

Understood The Software Requirements specifications for a Weather appliation.

# References

**Books:**

1. Roger S. Pressman, Software Engineering: A practitioners Approach, 7th Edition, McGraw Hill, 2010.
2. Technical report on Guidelines for Documents Produced by Student Projects In Software Engineering based on IEEE standards.
3. https://www.sharelatex.com/