**Software Requirements Specification**

for

**Wearther App**

Version 1.0 approved

Prepared by Bug Bytes

Southeast Missouri State

5/1/2024

|  |
| --- |

Table of Contents

Table of Contents........................................................................................................................... 2

Revision History............................................................................................................................. 3

1.   Introduction............................................................................................................................. 4

1.1   Purpose................................................................................................................................. 4

1.2   Document Conventions.......................................................................................................... 5

1.3   Intended Audience and Reading Suggestions......................................................................... 6

1.4   Project Scope......................................................................................................................... 7

1.5   References............................................................................................................................. 8

2.   Overall Description................................................................................................................. 9

2.1   Product Perspective................................................................................................................ 9

2.2   Product Features................................................................................................................... 10

2.3   User Classes and Characteristics........................................................................................... 11

2.4   Operating Environment......................................................................................................... 12

2.5   Design and Implementation Constraints............................................................................... 13

2.6   User Documentation............................................................................................................. 14

2.7   Assumptions and Dependencies........................................................................................... 14

3.   System Features...................................................................................................................... 15

3.1   System Feature 1................................................................................................................... 16

3.2   System Feature 2 (and so on) ............................................................................................... 17

4.   External Interface Requirements............................................................................................. 18

4.1   User Interfaces................................................................................................................. 18-20

4.2   Hardware Interfaces.............................................................................................................. 20

4.3   Software Interfaces............................................................................................................... 22

4.4   Communications Interfaces.................................................................................................. 23

5.   Other Nonfunctional Requirements........................................................................................ 24

5.1   Performance Requirements................................................................................................... 24

5.2   Safety Requirements............................................................................................................. 26

5.3   Security Requirements.......................................................................................................... 27

5.4   Software Quality Attributes.................................................................................................. 28

6.   Other Requirements................................................................................................................ 30

Appendix A: Glossary................................................................................................................... 30

Appendix B: Analysis Models...................................................................................................... 31

Appendix C: Issues List................................................................................................................ 31

Revision History

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

**1.**     **Introduction**

**1.1**     **Purpose**

1. **Weather Retrieval**:
   * The app's ability to fetch weather information is crucial for providing users with up-to-date data for their selected location. This includes both automatic retrievals based on the user's current location and manual search functionality.
2. **Weather Display**:
   * Once weather data is retrieved, the app should accurately display current weather conditions to users. This information serves as the basis for providing clothing suggestions and informing users about weather conditions in their area.
3. **Clothing Suggestions**:
   * A unique feature of the app is providing tailored clothing suggestions based on the current weather conditions. This functionality enhances user experience by offering practical advice and ensuring comfort and safety in different weather scenarios.
4. **User Interface**:
   * The design and layout of the app's interface play a crucial role in user engagement and satisfaction. Effective presentation of weather information and clothing suggestions is essential for ease of use and accessibility.
5. **Location Services**:
   * Utilizing location services allows the app to provide personalized weather information based on the user's current location. This enhances the app's convenience and relevance to users.
6. **Search Functionality**:
   * Users should have the flexibility to manually search for weather information for specific locations beyond their current location. This feature caters to users who may be planning trips or interested in weather conditions in different areas.
7. **Reliability and Accuracy**:
   * Ensuring the reliability and accuracy of weather information and clothing suggestions is paramount. Users rely on the app for making decisions related to outdoor activities and attire, making accuracy a critical aspect of the app's functionality.

**1.2**     **Document Conventions**

1. **Priority Levels**:

* Each requirement statement may have its own priority level assigned to it. This priority level indicates the importance or urgency of implementing that requirement. Priorities are typically denoted using standardized symbols or keywords, such as "High," "Medium," or "Low," or using numerical values (e.g., 1, 2, 3).

1. **Typographical Conventions**:
   * Bold or italic text may be used to highlight key terms, headings, or important points within the document.
   * Bullet points or numbered lists may be used to organize requirements, sub-requirements, or related information in a structured manner.
   * Tables or charts may be utilized to present complex data or relationships between requirements in a clear and concise format.
   * To Be Added or (TBA) should be displayed to allow readers to see future implementation, and current working points.
2. **Standardized Formats**:
   * Requirements may be structured using a standardized format or template, with sections dedicated to describing the requirement ID, description, rationale, acceptance criteria, and priority level.
   * Consistent terminology and language may be used throughout the document to ensure clarity and avoid ambiguity.
3. **Inheritance of Priority Levels**:
   * Higher-level requirements may implicitly inherit priority levels to their detailed requirements. For example, if a high-level requirement is deemed critical, its detailed sub-requirements may also be assigned high priority by default unless explicitly stated otherwise.

Alternatively, each requirement statement may have its own distinct priority level based on its individual importance and impact on the overall system functionality.

**1.3**     **Intended Audience and Reading Suggestions**

**Sequence for reading the document should be as follows:**

1. **Overview Sections:**
   * Start with the introduction and overview sections to gain a high-level understanding of the document's purpose, scope, and intended audience. This provides context for the rest of the document.
2. **Functional Requirements:**
   * Developers, testers, and project managers should focus on the functional requirements section to understand the app's core functionalities. This section outlines what the app should do and how it should behave in various scenarios.
3. **Non-Functional Requirements**:
   * Project managers, developers, and testers should review the non-functional requirements to ensure that the app meets performance, reliability, and usability standards. This section covers aspects such as performance, security, scalability, and accessibility.
4. **User Interface Requirements:**
   * Developers, designers, and testers can review the user interface requirements to understand the app's design and layout specifications. This section outlines how the app should look and feel to users, including interface elements, navigation, and visual design guidelines.
5. **External Interface Requirements:**
   * Developers and testers should review this section to understand how the app interacts with external systems or services. This includes APIs, integrations, data exchanges, and communication protocols.
6. **System Requirements:**
   * Developers, project managers, and testers should review the system requirements to ensure that the app can be properly installed and run on various devices. This section covers hardware, software, network, and other infrastructure requirements.
7. **Quality Assurance and Testing:**
   * Testers and quality assurance professionals should focus on this section to understand the testing requirements and guidelines. This includes test plans, test cases, test data, and testing methodologies to ensure the app meets quality standards.
8. **Documentation Requirements:**
   * Documentation writers should review this section to understand the requirements for creating user manuals, guides, and other documentation related to the app. This includes content, format, delivery, and maintenance of documentation throughout the app's lifecycle.

**1.4**     **Project Scope**

**1. Benefits, Objectives, and Goals:**

* **Real-Time Updates**: The primary objective is to provide up-to-date weather information to help users plan their daily activities safely and efficiently. This ensures that users have access to accurate data when making decisions about their plans.
* **User Engagement**: Another goal is to increase user engagement by allowing users to create accounts, log in, and explore weather updates as guests. This strategy aims to foster a wider user base and encourage repeat usage of the application.
* **Scalability for Future Expansion**: Establishing a solid foundation for potential expansion into mobile platforms is a key objective. This approach ensures that the application can reach a broader audience and integrate more advanced features in future releases, supporting long-term growth and adaptability.

**2. Relation to Corporate Goals:**

* The development of this weather application aligns with corporate goals of enhancing user engagement and expanding technological footprint. By providing a reliable and accessible digital service, the aim is to establish the brand as a leader in digital solutions, enhancing customer satisfaction and loyalty.
* This project also sets the stage for future growth into mobile application markets, supporting the strategy to capture and engage a mobile-first audience.

**3. Scope of Current Release:**

* **Deploying accurate and efficient location-based weather tracking**: The focus is on delivering reliable weather tracking functionality that provides precise and timely updates based on the user's location.
* **Offering a clean, intuitive web interface for accessing weather data**: The goal is to create an easy-to-use interface that allows users to access weather information seamlessly.
* **Preparing the architecture for future adaptations and expansions into mobile app development**: The emphasis is on building a scalable architecture that can support future expansi--ons into mobile platforms. This includes laying the groundwork for integrating advanced features and accommodating evolving user needs.

This release prioritizes leveraging current technological capabilities to deliver an effective and engaging user experience while also preparing for future enhancements based on user feedback and market trends.

**1.5**     **References**

* **Title:** Functional Requirements/Non-Functional Requirements (PD1\_FRNFR)
  + **Author:** Bug Bytes Team
  + **Version:** 1.3
  + **Latest update:** 5/3/2024
  + **Source:** <https://github.com/eye-zack/CS445-Group-Project/blob/main/PD1/PD1_FRNFR.docx>
* **Title:** Use Cases
  + **Author:** Bug Bytes Team
  + **version:** 1.2
  + **Latest update:** 2/3/2024
  + **Source:** <https://github.com/eye-zack/CS445-Group-Project/blob/main/PD2/Revised_PD1_Uses_Case_based_on_your_Feedback.docx>
* **Title:** Use Case Diagram
  + **Author:** Bug Bytes Team
  + **Version:** 1.0
  + **Latest Update:** 3/3/2024
  + **Source:** <https://github.com/eye-zack/CS445-Group-Project/blob/main/PD2/Use_case_diagram_1.pdf>
* **Title:** Weather app template
  + **Author:** Bug Bytes Team
  + **Version:** 1.0
  + **Latest update:** 2/3/2024
  + **Source:** <https://github.com/eye-zack/CS445-Group-Project/blob/main/PD1/weather%20app%20template.png>
* **Title:** Initial Class Diagram
  + **Author:** Bug Bytes Team
  + **Version:** 1.2
  + **Latest Update:** 5/3/2024
  + **Source:** <https://github.com/eye-zack/CS445-Group-Project/blob/main/PD2/Initial_ClassDiagram.svg>
* **Title:** State Diagram
  + **Author:** Bug Bytes Team
  + **Version:** 1.2
  + **Latest Update:** 5/3/2024
  + **Source:** <https://github.com/eye-zack/CS445-Group-Project/blob/main/PD3/PD3%20Diagrams/StateDiagramV1.2.svg>
* **Title:** Location based services (Zip Code Based Location)
  + **Author:** Bug Bytes Team
  + **Version:** 1.0
  + **Latest Update:** 4/12/2024
  + **Source:** <https://github.com/eye-zack/CS445-Group-Project/blob/main/PD3/PD3%20Diagrams/Zipcode%20Based%20Location.svg>

**2.** **Overall Description**

**2.1**     **Product Perspective**

Wearther App represents a new take on web-based weather applications, designed to provide users with timely and accurate weather information. This product is not a follow-on member of a product family nor a replacement for an existing system. It introduces innovative features such as personalized clothing suggestions and a user-centered design, distinguishing it in the digital weather services market.

Wearther App is not a self-contained system as it relies on external weather data APIs for its core functionalities, positioning it within a broader ecosystem of data providers and consumer applications.

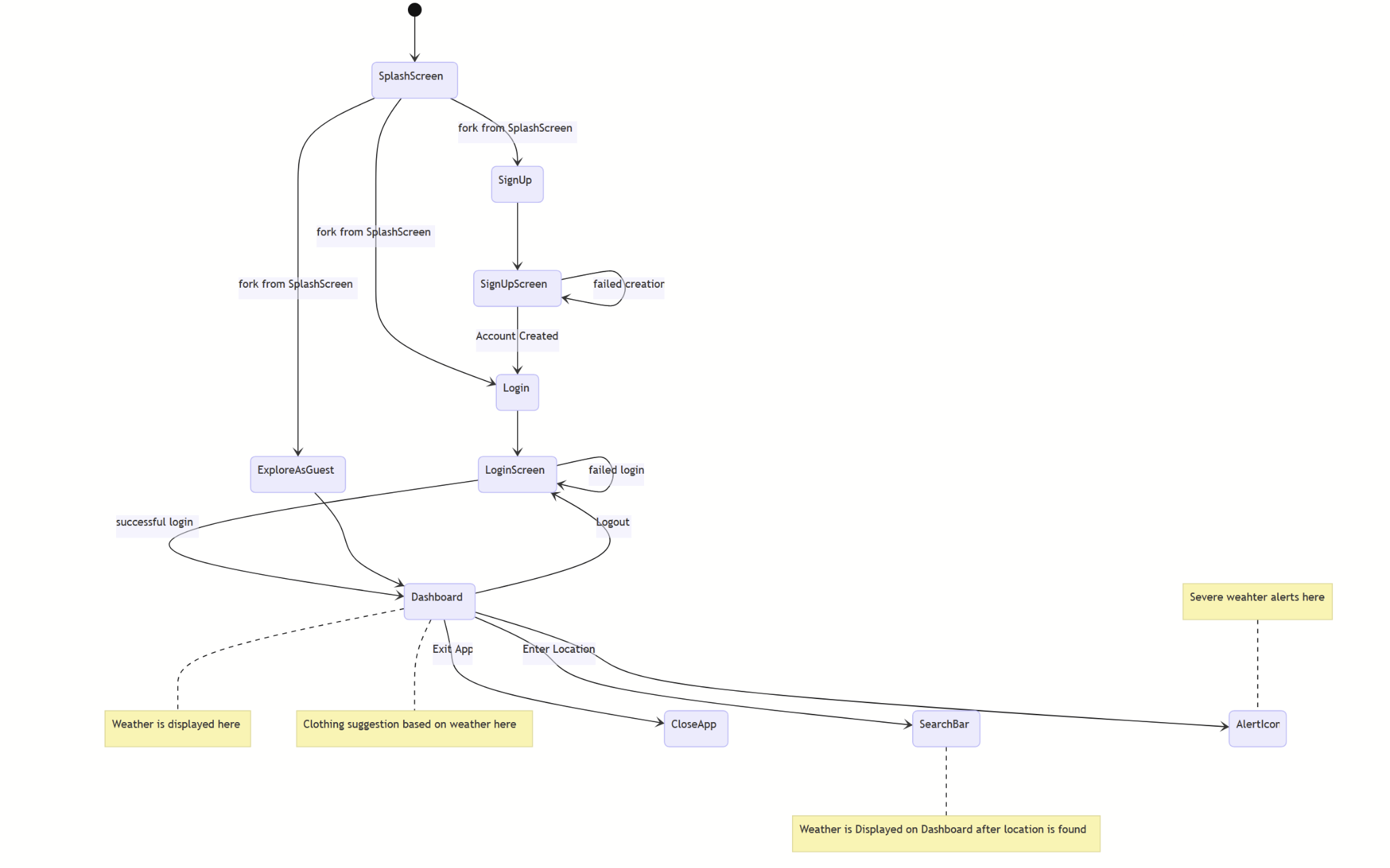
Wearther App integrates with external APIs to retrieve essential weather data. These APIs are critical as they supply real-time weather information based on user locations or search inputs. This integration makes the app dependent on external services for its core functionality, which includes the following processes:

* **Data Retrieval:** The app fetches weather data from external APIs based on user inputs. This involves sending queries to these APIs which respond with the relevant weather data.
* **Data Processing:** After retrieving the data, Weather App analyzes and processes the weather information to present it in a user-friendly format. This step transforms raw data into easily understandable metrics and integrates this information into the broader context of the app's functionalities.
* **Data Display**: The final processed data is displayed through the app’s user interface. This involves not just showing the current weather but also using this data to generate relevant user content such as clothing suggestions.

**2.2**     **Product Features**

Wearther is a web-based weather application that provides users with real-time, location-based weather information alongside personalized clothing recommendations to help users dress appropriately according to the weather conditions. The key features of Wearther are organized into several functional groups:

* **Location Based Weather Information:**
  + Retrieves and displays real-time weather data based on the user’s current location or a manually entered location.
  + Shows details such as temperature, humidity, wind speed, and precipitation.
* **Clothing Recommendations:**
  + Provides clothing suggestions based on the current weather conditions to help users make appropriate attire decisions.
* **User Interactions:**
  + Allows users to either utilize location services automatically or input locations manually for weather checks.
  + Offers a simple, intuitive user interface for navigating between weather information and clothing recommendations.
* **Data Security and Privacy:**
  + Ensures secure data transmission using HTTPS, protecting users’ location data.
* **Error Handling and User Notifications:**
  + Manages errors effectively, such as location services being disabled or issues with data retrieval from the weather API.
  + Displays user-friendly error messages and provides options to retry operations or enter information manually.



**2.3**     **User Classes and Characteristics**

1. Casual Users:

* **Frequency of Use:** Occasional - typically during specific weather conditions or prior to outdoor activities.
* **Product Functions Used:** Generally uses the app for immediate, current weather updates and clothing suggestions.
* **Technical Expertise:** Low, this group values straightforward and easy-to-navigate interfaces.
* **Characteristics:** These users may have varying levels of education and are primarily looking for quick and easy guidance on how to dress for the day’s weather. They seek quick, hassle-free information on weather conditions to help them decide how to dress appropriately for the day.

1. Commuters:

* **Frequency of Use:** High during weekdays, especially during commuting hours.
* **Product of Functions Used:** Primarily interested in up-to-minute weather forecasts to make informed decisions about travel routes and appropriate clothing.
* **Technical Expertise:** Moderate; comfortable with basic app functionalities but not looking for complex configuration options.
* **Characteristics:** Typically professionals or students who rely on timely weather updates to avoid commuting delays and to ensure comfort and safety during travel.

**Importance of User Classes**

* **Primary Users:** The design and functionality of Wearther are heavily influenced by the needs of casual users and commuters, who form the bulk of the app’s user base. These classes are vital as they directly affect the core operational focus of the app, ensuring it remains relevant and highly usable on a day-to-day basis.
* **Priority level:** Casual users and commuters are given the highest priority in terms of feature development and user experience considerations. This approach helps maintain high engagement levels and satisfaction among the most frequent users of the app.

**2.4**     **Operating Environment**

1. **Hardware Platform**

* The application is designed to be accessed via the web browsers on any device capable of running them, including desktops and laptops.

1. **Operating Systems**

* Wearther is platform-agnostic and can be accessed on any operating system with a modern web browser, including windows, macOS, and Linux.

1. **Web Browsers**

* The application is compatible with the latest versions of major web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. Users should ensure they are using recent versions of these browsers for the best experience.

1. **Network Requirements**

* Requires a stable internet connection for optimal performance, but no specific minimum speed is mandated, as performance will scale with available bandwidth.

**2.5**     **Design and Implementation Constraints**

1. **Corporate/Regulatory Policies:**

* **Data Privacy:** Compliance with General Data Protection Regulation (GDPR), California Consumer Privacy Act (CCPA), and other relevant data protection regulations affecting the handling or user data.
* **Accessibility Standards:** The application must adhere to Americans with Disabilities Act (ADA) compliance standards to ensure accessibility for all users.

1. **Client Performance:**

* **Optimization:** The application should be optimized for low memory usage and efficient processing to accommodate users with older or less powerful devices. This includes optimizing and minimizing the size of assets delivered to the client.
* **Responsive Design:** Ensure that the user interface remains responsive across various resolutions, adapting layout and functionality to both high and low-end devices.

1. **Network and Data Processing:**

* **Latency:** Strive for minimal latency in data processing and content delivery to ensure that the application loads quickly even on slower internet connections. Techniques such as effective use of Content Delivery Network (CDNS), caching strategies, and data compression should be employed.
* **Real-Time Data Interaction:** While specific hardware limitations are not a primary constraint, testing should include scenarios that simulate lower-end hardware. This can include throttling CPU in browser developer tools to simulate slower processing speeds and using tools to limit available Random Access Memory (RAM).

1. **Specific Technologies, Tools, Database**

* **Frontend:** Development restricted to React.js to align with existing project infrastructure.
* **Backend:** Node.js for server side operation
* **Database:** Scalability with the use of MongoDB considering its scalability and flexibility (TBD).

1. **Security Considerations:**

* **Data Transmission:** All data exchanges between the client and server must be secured using HTTPS, incorporating TLS encryption to safeguard data integrity and privacy.
* **Data Storage:** Sensitive data stored by the application must be encrypted at rest, using industry standard encryption protocols.

1. **Development and Maintenance:**

* **Code Standards and Convention:** Development will adhere to well-established JavaScript style guidelines to ensure code quality and consistency. This approach will help in maintaining a clean, readable, and maintainable code base.
* **Documentation:** Comprehensive documentation is required for all aspects of the application, facilitating future maintenance and scalability considerations.

**2.6**     **User Documentation**

To ensure that Wearther is accessible and user-friendly, the following user documentation will be developed and delivered alongside the software.

1. **Quick Start Guide**

* A concise document designed to help new users begin using the application effectively in the shortest possible time. It will cover the basic functions like logging in, account creation, viewing the weather forecast and setting up weather notifications.

1. **Online Help**

* An integrated help system within the application that offers quick tips, FAQs, and step-by-step guide for common tasks. This system will be context-sensitive, providing relevant help based on the user’s current actions and screen.
* Implementation will be as a searchable help section within the application, featuring pop-up help dialogues and dedicated help menu.

**2.7**     **Assumptions and Dependencies**

**Assumptions**

1. Third-Party Weather API

* The weather data for Wearther will be sourced from third-party APIs, assumed to provide timely, accurate, and reliable data.
* If these APIs failed to deliver as expected it could affect the functionality of the application.

1. User Device Compatibility

* Users will access the application on devices equipped with modern web browsers that are capable of handling advanced JavaScript and CSS.
* If users access the application with outdated or unsupported browsers, they may experience reduced functionality or performance issues.

1. Internet Connectivity

* Users will have stable and continuous access to the internet when using Wearther.
* Unreliable internet access could impair the real-time functionality of the application, affecting user satisfaction.

1. Development Environment

* Ongoing development of the application will use specified versions of software development tools and environments (e.g., specific versons of Node.js and React).
* Changes or updates in these tools could require unexpected adaptations in the development process.

**Dependencies**

1. Compliance and Regulatory Requirements

* The application is assumed to comply with relevant data privacy laws that apply to user data handling.
* Any misalignment with these regulations could lead to legal penalties and damage to reputation.

1. Hosting Services

* The application is planned to be hosted on cloud platforms that promise scalability and reliability.
* Potential disruptions in cloud services or unexpected changes in service policies could impact the availability and performance of Wearther.

**3.** **System Features**

**3.1**     **Location-Based Weather Display**

3.1.1 Description and Priority

This feature enables the application to display real-time weather information based on the user’s current location. It is considered a high priority feature due to its direct impact on user satisfaction and engagement.

* **Benefit: 9** - It is essential for user engagement and usability.
* **Penalty: 9** - Lack of this feature would severely diminish user experience.
* **Cost: 3** - Not resource intensive to achieve user location.
* **Risk: 3** - Low technical risk due to well-established methods for location tracking

3.1.2 Stimulus/Response Sequences

* **Stimulus:** The user opens the application.
* **Response**: Application attempts to retrieve location
  + If location services are not enabled, an error message is displayed.
  + If location is enabled, this step is skipped.
* **Simulus**: The user enables location services (if they were not already enabled).
* **Response:** The application retrieves the user’s geographical coordinates after location services are confirmed to be enabled.
* **Stimulus:** The application has geographical coordinates.
* **Response:** The application sends a request to the weather API using the geographical coordinates.
* **Stimulus:** The weather API receives the request.
* **Response**: The weather API returns the current weather data for the user’s location.
* **Stimulus**: The user views the weather data displayed on the app.
* **Response**: The application updates the display when the user requests a refresh.

3.1.3 Functional Requirements

* **REQ-1:** The application must accurately determine the user’s current location using the device’s location services.
* **REQ-2:** The application must securely transmit the location data to the weather information API using standard HTTPS protocols.
* **REQ-3**: The application shall display the current temperature, humidity, wind speed, and precipitation information.
* **REQ-4:** The application must update the weather information upon user request.
* **REQ-5:** The application should handle errors in location services by prompting the user to enable these services or enter a location manually.
* **REQ-6:** The application shall handle API errors or data retrieval failures by displaying a user-friendly error message.
* **REQ-7:** The system must ensure that all data transmissions to and from the weather API are conducted over HTTPS to utilize TLS encryption, aligning with standard internet security practices.

**3.2**     **Clothing Recommendation Based on Weather**

3.2.1 Description and Priority

This feature enables the application to suggest appropriate clothing options to users based on the current and forecasted weather conditions at their location. It is a high-priority feature as it directly enhances the utility of the app, making it not only a tool for weather updates but also a daily planning aid.

* **Benefit:** **9** - Significantly increases user engagement by integrating practical daily-use scenarios.
* **Penalty: 7** - Without this feature, the app would be less appealing compared to other apps.
* **Cost: 4** - Involves data analysis which may require advanced algorithms based on application growth.
* **Risk: 3** - The risk is relatively low as the implementation relies on existing weather forecast data and basic logic for clothing recommendations.

3.2.2 Stimulus/Response Sequences

* **Stimulus:** User enters or confirms their location.
* **Response:** The application retrieves the latest weather forecast for that location.
* **Stimulus:** The forecast data is available
* **Response:** Based on the weather conditions, the application suggests appropriate clothing options.
* **Stimulus:** User views the daily weather forecasts
* **Response:** Alongside the weather details, clothing recommendations are displayed, advising the user on what to wear.

3.2.3 Functional Requirements

* **REQ-1:** The application must accurately fetch the weather forecast for the user’s specified location.
* **REQ-2:** The system shall analyze the forecasted weather conditions to determine suitable clothing recommendations.
  + Acceptance Criteria: The recommendations should align with the anticipated weather.
* **REQ-3:** The application must display these recommendations in a user-friendly format alongside the weather forecast.
* **REQ-4:** The application must handle scenarios where forecast data might be incomplete or updating, by displaying a relevant message or default recommendations.
* **REQ-6:** The application should ensure that recommendations are updated whenever there is a significant change in the forecast.

**4.**     **External Interface Requirements**

**4.1** **User Interfaces**

The user interface (UI) of the Wearther app plays a crucial role in providing users with an intuitive and engaging experience while accessing weather information and clothing suggestions. Here are the logical characteristics of each interface between the software product and the users:

1. **Main Screen Interface**:
   * The dashboard presents the current weather conditions for the user's selected location after successful login.
   * It displays essential weather data such as temperature, humidity, wind speed, and precipitation.
   * The dashboard provides easy weather readability along with convenient clothing suggestions.
2. **Search Functionality Interface**:
   * Users can access the search functionality to manually search for weather information for specific locations.
   * The search interface allows users to enter location names to retrieve weather forecasts.
   * Autocomplete suggestions may enhance user experience and speed up the search process.
3. **Clothing Suggestions Interface**:
   * Upon retrieving weather data, the app provides clothing suggestions tailored to the current weather conditions.
   * The interface displays recommended clothing items based on factors like temperature, precipitation, and wind speed.
4. **Settings Interface**:
   * Currently the settings are fixed.
   * (TBA)The settings interface allows users to customize app preferences such as temperature units (Celsius or Fahrenheit) and language settings.
   * (TBA)Users can also manage location preferences, enable/disable notifications, and adjust accessibility options.
5. **Error Handling Interface**:
   * When errors occur, the app should display clear and informative error messages.
   * Error messages should be presented prominently on the screen and include actionable steps for users to take.
6. **Help and Support Interface**:
   * (TBA)Users can access help and support resources from within the app, including FAQs, troubleshooting guides, and contact information for customer support.
   * (TBA)A help button or menu option should be readily accessible from any screen within the app.
7. **Consistent Design Elements**:
   * The app follows a consistent design language and layout across all screens to ensure a cohesive user experience.
   * Standard buttons and functions, such as navigation menus appear consistently throughout the app.
   * Visual elements adhere to established design standards, including color schemes, typography, and iconography.
8. **Accessibility Interface**:
   * The app incorporates accessibility features to ensure usability for all users, including those with disabilities.
   * (TBA)Accessibility options may include screen reader support, text resizing, color contrast adjustments, and keyboard navigation shortcuts.

These logical characteristics define the various interfaces between the Wearther app and its users, ensuring a user-friendly, accessible, and consistent experience across all interactions. Detailed design specifications for each interface are documented in a separate user interface specification document.

**4.2**     **Hardware Interfaces**

The Wearther app interfaces with hardware components to deliver weather information and user interactions effectively. Here are the logical and physical characteristics of each interface between the software product and the hardware components of the system:

1. **Supported Device Types**:
   * The app currently supports desktop computers.
   * (TBA) It is designed to be responsive and adaptable to different screen sizes and resolutions, ensuring a consistent user experience across devices.
2. **Data and Control Interactions**:
   * The app interacts with hardware components to retrieve location data for the user's current location.
   * It accesses device location services to determine users geolocational positioning.
   * User interactions, such as keyboard inputs, and mouse clicks, are captured and processed by the app to navigate through screens, input text, and trigger actions.
3. **Communication Protocols**:
   * The app communicates with external servers and APIs using standard protocols such as HTTP/HTTPS.
   * It retrieves weather data from online sources and sends user queries for location-based weather forecasts.
   * Secure communication protocols, such as SSL/TLS, are employed to ensure data privacy and integrity during transmission.
4. **Device Compatibility**:
   * The app is compatible with a wide range of operating systems such as Windows, Linux and macOS.
   * It utilizes platform-specific APIs and frameworks to access hardware features and ensure optimal performance on each supported platform.
5. **Network Connectivity**:
   * The app requires internet connectivity to fetch weather data and communicate with external servers.
   * It supports both Wi-Fi and cellular network connections, adapting to the available network environment for seamless operation.
6. **Offline Functionality**:
   * Currently the app only functions while online. If the app loses connection to an internet provider the app will only display last known data.
   * (TBA)Cached data may be utilized to display previously retrieved weather forecasts and clothing suggestions, ensuring a degree of functionality in offline scenarios.
7. **Hardware Resource Utilization**:
   * The app optimizes CPU usage, and memory footprint, enhancing device performance and user experience.
   * Background processes and push notifications are managed efficiently to avoid draining device resources unnecessarily.

These logical and physical characteristics define how the Wearther App interacts with hardware components, ensuring seamless integration, optimal performance, and compatibility across a diverse range of devices and platforms.

**4.3**     **Software Interfaces**

1. **Weather Data API**:
   * The app connects to a weather data API (OpenWeather API, Pirate Weather API) to retrieve current weather conditions and forecasts for specified locations. I mention both because there is a (TBA) for a future update.
   * **Data item (City Name) incoming to the system**: Location coordinates or names for weather data retrieval.
   * **Data items outgoing from the system:** Current weather conditions, forecasts, and related metadata.
   * **Purpose**: Fetches accurate and up-to-date weather information for display to users.
2. **Geolocation Services**:
   * The app utilizes geolocation services provided by the operating system to determine the user's current location.
   * **Data items incoming to the system**: Device location data obtained from desktop location services.
   * **Data items outgoing from the system**: User's current geographical coordinates.
   * **Purpose:** Determines the user's location for retrieving localized weather data.
3. **User Authentication and Management System**:
   * The app integrates with a user authentication and management system to handle user registration, login, and account management functionalities.
   * **Data items incoming to the system**: User credentials (e.g., username, password, email).
   * **Data items outgoing from the system**: Authentication tokens, user profile information.
   * **Purpose**: Manages user accounts and ensures secure access to personalized features.
4. **Local Storage**:
   * The app utilizes local storage mechanisms provided by the operating to cache weather data, user preferences, and application state.
   * **Data items incoming to the system**: Weather data, user preferences.
   * **Data items outgoing from the system:** Cached weather data, user settings.
   * **Purpose:** Stores frequently accessed data locally to improve app performance
5. **(TBA) Push Notification Service**:
   * The following content below details a future update.
   * The app integrates with a push notification service to deliver real-time weather alerts and updates to users.
   * **Data items incoming to the system:** Weather alerts, notifications.
   * **Data items outgoing from the system:** Push notification messages.
   * **Purpose:** Notifies users of important weather events and updates in real-time.

These software components enable the Weather app to deliver personalized weather information, user authentication, local storage, push notifications, analytics, and error monitoring functionalities. The communication between these components occurs via defined APIs, libraries, and operating system interfaces, ensuring seamless integration and efficient data exchange. Detailed API protocols and implementation constraints may be documented in separate technical specifications or developer documentation.

**4.4**     **Communications Interfaces**

1. **API Communication Protocols**:
   * The app communicates with external weather data APIs (e.g., OpenWeather API, Pirate API) using standard HTTP or HTTPS protocols.
   * **Message Formatting**: Requests and responses adhere to JSON following our API specifications.
   * **Communication Standards:** HTTP/HTTPS for data retrieval and transmission.
   * **Security and Encryption:** All API communications are encrypted using SSL/TLS protocols to ensure data privacy and integrity.
   * **Data Transfer Rates:** Requests and responses should be optimized for efficient data transfer, with consideration for bandwidth limitations and network latency.
   * **Synchronization Mechanisms:** The app may implement caching mechanisms to minimize API calls and synchronize data updates in the background.
2. **Geolocation Services**:
   * The app communicates with the device's operating system to access geolocation services for determining the user's current location.
   * **Communication Standards:** Platform-specific location API (rapid API) handle communication with the operating system.
   * **Security and Encryption**: Location data transmission is secured through built-in platform security features.
   * **Data Transfer Rates**: Geolocation updates are requested at appropriate intervals to balance accuracy with battery consumption.
   * **Synchronization Mechanisms**: The app may implement background location updates to maintain up-to-date location information.
3. **Network Connectivity**:
   * The app communicates with external servers and APIs to retrieve weather data and other information.
   * **Communication Standards**: Standard TCP/IP protocols are used for network communication.
   * **Security and Encryption**: All network communications are encrypted using SSL/TLS to protect against data interception and tampering.
   * **Data Transfer Rates:** Data transfer rates are optimized to minimize latency and maximize throughput for efficient data retrieval.

These communication requirements ensure that the Wearther app can effectively retrieve weather information, determine user locations, deliver push notifications, and maintain network connectivity while adhering to established standards for security, encryption, data transfer rates, and synchronization.

**5.**     **Other Nonfunctional Requirements**

**5.1**     **Performance Requirements**

**Here are the performance requirements outlined for different features of the Wearther App:**

* 1. **Weather Retrieval:**
* The app must retrieve weather information within seconds of initiating a request.
  + Rationale: Users expect timely access to weather data, and a delay of more than a few seconds could lead to frustration and poor user experience.
  1. **Weather Display:**
* The app should render weather information on the user interface within a few seconds of receiving the data.
  + Rationale: Quick display of weather information ensures that users receive real-time updates without significant delays.
  1. **Clothing Suggestions:**
* The app should generate clothing suggestions within seconds of receiving weather data.
  + Rationale: Users rely on the app for timely recommendations on what to wear, especially when planning their activities for the day.
  1. **Search Functionality:**
* The app should display search results for user-entered locations within seconds of initiating a search.
  + Rationale: Users expect quick responses when searching for weather information for specific locations.
  1. **Real-Time Updates (for real-time systems):**
* The app should provide real-time updates of weather conditions every 30 minutes.
  + Rationale: Real-time updates ensure that users have the most current weather information available.
  1. **Location Services:**
* The app should accurately determine the user's location within a margin of error of 100 meters.
  + Rationale: Accurate location detection is essential for providing relevant weather information based on the user's current location.
  1. **Response Time:**
* All user interactions within the app should result in a response time of less than seconds.
  + Rationale: Responsiveness is critical for providing a seamless and engaging user experience.

**5.2**     **Safety Requirements**

**For the Wearther App, safety requirements are primarily concerned with ensuring that users are not exposed to any risks or harm while using the application. While the app itself may not pose direct physical risks, there are still considerations related to data privacy, accuracy of weather information, and user interactions. Here are some safety requirements and related actions:**

1. **Data Privacy and Security:**

* **Requirement:** All user data collected by the app, including location information, must be securely stored and transmitted.
  + **Safeguard**: Implement encryption protocols to protect user data during transmission and storage.
  + **Prevention**: Ensure that user data is not shared with third parties without explicit consent and is only used for providing weather-related services.

1. **Accuracy of Weather Information:**

* **Requirement:** The app must provide accurate and reliable weather forecasts to prevent users from making decisions based on incorrect information.
  + **Safeguard**: Regularly update weather data sources and implement quality control measures to verify forecast accuracy.
  + **Prevention:** Clearly communicate to users that weather forecasts are predictions and may not always be 100% accurate.

1. **User Interface Safety:**

* **Requirement:** The user interface should be designed to prevent accidental inputs or actions that could lead to undesired outcomes.
  + **Safeguard:** Implement confirmation dialogs for critical actions, such as deleting user data or settings.
  + **Prevention**: Design user interface elements, such as buttons and menus, to be easily distinguishable and require intentional user interaction.

1. **Compliance with Privacy Regulations:**

* **Requirement**: Ensure compliance with relevant data privacy regulations.
  + **Safeguard**: Implement mechanisms for users to control their data and provide transparency about data collection and usage practices.
  + **Prevention**: Regularly review and update privacy policies to align with evolving regulatory requirements.

1. **Safety Certifications:**

* **Requirement**: Obtain relevant safety certifications, if applicable, to demonstrate compliance with industry standards and regulations.
  + **Safeguard:** Define any specific certifications required by regulatory bodies or industry associations.
  + **Prevention:** Ensure that the app adheres to safety guidelines specified by relevant authorities.

**5.3**     **Security Requirements**

1. **Data Privacy and Protection**:
   * User data, including location information and preferences, must be securely stored, and transmitted to prevent unauthorized access or disclosure.
   * **Encryption of Data**: All sensitive data transmissions between the app and external servers must be encrypted using SSL/TLS protocols to prevent eavesdropping and data interception.
2. **User Identity Authentication**:
   * User authentication is required for accessing personalized features such as saved locations, preferences, and notifications.
   * **Strong Authentication Measures**: Implement secure authentication mechanisms, such as password-based authentication.
3. **User Consent and Transparency**:
   * Users must explicitly consent to the collection and processing of their personal data for the app's functionality.
   * **Transparent Data Usage:** Provide clear and concise privacy policies and terms of service that outline how user data is collected, used, and shared within the app.
   * **Opt-In/Opt-Out Mechanism**: Allow users to opt in or opt out of certain data collection and usage practices, such as location tracking or personalized recommendations.
4. **Third-Party Integrations**:
   * Ensure that third-party integrations, such as weather data APIs and push notification services, comply with security and privacy standards.
   * **Data Sharing Agreements**: Establish data sharing agreements with third-party providers to ensure that user data is handled securely and used only for authorized purposes.
5. **Security Audits and Assessments**:
   * Conduct regular security audits and assessments to identify and address potential vulnerabilities in the app's infrastructure and codebase.
6. **Compliance with Security Standards**:
   * Adherence to Industry Regulations: Ensure compliance with industry-specific security regulations and standards applicable to weather forecasting and mobile applications.

By adhering to these security and privacy requirements, the Wearther app can establish trust with users, protect sensitive data, and mitigate the risk of security breaches or privacy violations.

**5.4**     **Software Quality Attributes**

1. **Usability**:
   * Customers prioritize ease of use over complexity. The app should have a user-friendly interface with intuitive navigation and clear instructions.
   * **Verifiable Metric:** Measure user satisfaction through surveys or usability testing, aiming for a satisfaction score of 90% or higher.
2. **Reliability**:
   * Users rely on the app for accurate weather information. The app should consistently provide reliable data without downtime or service disruptions.
   * **Verifiable Metric**: Calculate the app's uptime percentage, aiming for at least 99.9% availability over a specified period.
3. **Maintainability**:
   * Developers should be able to efficiently update and maintain the app's codebase. This includes well-structured code, clear documentation, and modular design.
   * **Verifiable Metric**: Measure the time and effort required to implement updates or bug fixes, aiming for a decrease in maintenance time over successive releases.
4. **Portability**:
   * Users may access the app from various operating systems and browsers. The app should be compatible and perform consistently across different operating systems and web browsers.
   * **Verifiable Metric**: Test the app on multiple browsers and operation systems, ensuring consistent functionality and appearance across all supported environments.
5. **Testability**:
   * Developers need robust testing capabilities to identify and fix software defects. The app should be designed for easy testing, with comprehensive test coverage and automation support.
   * **Verifiable Metric**: Measure the percentage of code coverage achieved by automated tests, aiming for 80% or higher coverage.
6. **Adaptability**:
   * The app should adapt to changing user needs and technological advancements. This includes support for new features, customization options, and integration with emerging technologies.
   * **Verifiable Metric**: Monitor user feedback and adoption rates for new features, aiming for positive trends in user engagement and satisfaction.
7. **Interoperability**:
   * The app should seamlessly integrate with other software components and services, such as location services and push notification systems.
   * **Verifiable Metric**: Test interoperability with external systems, ensuring smooth data exchange and functionality without errors or conflicts.
8. **Robustness**:
   * The app should gracefully handle unexpected inputs, errors, and adverse conditions without crashing or compromising functionality.
   * **Verifiable Metric**: Conduct stress testing and error simulation to assess the app's resilience under challenging conditions, aiming for stable performance under stress.

These quality characteristics ensure that the Wearther app meets the needs and expectations of both customers and developers, providing a reliable, user-friendly, and adaptable experience. Quantitative metrics and verifiable criteria enable continuous improvement and quality assurance throughout the app's lifecycle.

**6.**   **Other Requirements**

**Database Requirements**

* **Scalability:** The database must support scaling to handle increasing data volumes as user numbers grow (TBA).
* **Backup and Recovery:** Regular backups and robust recovery solutions must be implemented to prevent data loss (TBA).
* **Security:** Data must be encrypted both in transit and at rest, with access controlled through authentication and authorization mechanisms (TBA).

**Internationalization Requirements:**

* **Language Support:** The application must support multiple languages, starting with English and expanding into Spanish, French, and further international languages. (Currently English is the only language implemented).

**Legal Requirements:**

* **Compliance:** The application must comply with all relevant laws and regulations regarding user data protection and accessibility standards.

**Appendix A: Glossary**

**Wearther(App):** The software system described by this SRS.

**API (Application Programming Interface):** Set of routines, protocols, and tools for building software applications.

**GDPR (General Data Protection Regulation):** European Union Regulation for data protection and privacy.

**CCPA (California Consumer Privacy Act):** Legislation that provides California residents with the right to know what personal data is being collected about them, the right to delete personal data, and the right to opt-out of the sale of their personal data.

**ADA (Americans with Disabilities Act):** Legislation requiring accessibility for all American services and products.

**TBA (To Be Added)**: Placeholder for future updates, and necessary improvements.

**Appendix B: Analysis Models**

For any pertinent models or diagrams, [see section 1.5 references](https://docs.google.com/document/d/1xTHrOV5JExkMP1vfkbNJ308buYMSVewpw-MdWS7mSMs/edit#bookmark=id.z2tucm1nvqol).

**Appendix C: Issues List**

1. **Database Requirements:**

**Scalability:**

* + Decision pending on specific strategies and technologies for database scaling to accommodate increasing user numbers.

**Backup and Recovery:**

* + Finalize specific backup intervals and recovery solutions.

**Security:**

* + Determine encryption methods and authentication mechanisms.

1. **API Integration:**

**Weather Data API Selection:**

* + Final decision pending between OpenWeather API and Pirate Weather API based on cost, reliability, and data availability criteria.

1. **Supported Device Types:**

**Responsive Design Implementation:**

* + Pending specifics on how the app will adapt to different screen sizes and resolutions.

1. **Accessibility Interface:**

**Accessibility Features Implementation:**

* + Further details required on implementing screen reader support, text resizing, color contrast adjustments, and keyboard navigation shortcuts.

1. **Help and Support Interface:**

**Development of Help and Support Resources:**

* + Content for FAQs, troubleshooting guides, and customer support contract information needs development.

**Accessibility of Help Resources:**

* + Finalize design and placement of a help button or menu option accessible from any screen within the app.

1. **Settings Interface:**

**Customization of Settings:**

* + Development plans needed for allowing users to customize app preferences, such as temperature units and language settings**.**

**Location and Notification Settings:**

* + TBD implementation details for managing location preferences and enabling/disabling notifications.

1. **Technology and Tools:**

**Frontend and Backend Technologies:**

* + Further specification and testing protocol required for integration and detailed implementation using React.js for frontend and Node.js for backend.

1. **Database Choice and Configuration:**
   * MongoDB use and configuration details still under review, particularly concerning scalability and flexibilit**y.**