SOFTWARE REQUIREMENTS SPECIFICATION

For

Weather App using Database

Prepared by:-

Team : Enemy Of Errors

- Rackshana.B
- Bharath.P
- Lokesh.K

1. Introduction

1.1 Purpose

This document outlines the essential requirements for the Weather App with Database project. It aims to define both functional and non-functional aspects specified by the client. The primary purpose of the project is to create a user-friendly platform for accessing and managing weather data for various locations. The application's goal is to facilitate seamless interaction, data storage, and retrieval, offering users detailed weather information and reports. The project also emphasizes the utilization of ER diagrams and UML diagrams to illustrate hardware and software interfaces.

1.2 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

1.2 Scope of Development Project

The Weather App With Database offers users comprehensive weather information encompassing Temperature, Humidity, Pressure, Visibility, Wind, DewPoint, UV Index, Moon Phase, and Air Quality.

Users can input their preferred location (city) and seamlessly access real-time weather details for a particular week. The app's scope extends to providing a user-friendly interface for easy interaction, emphasizing simplicity and efficiency

Through a secure login system, users can customize their experience, ensuring individual preferences are met. The application aims to enhance user convenience by prioritizing realtime updates, empowering users with accurate and timely weather insights.

The absence of an administrative role streamlines the user experience, focusing on delivering precise and relevant weather data. The app's robust scope envisions providing users with a reliable and accessible tool for staying informed about current and upcoming weather conditions.

The Weather App With Database is designed to be adaptable and easily implementable across diverse scenarios. Its modular architecture allows for the seamless addition of new features as needed, promoting reusability and ensuring flexibility in accommodating evolving user requirements. The chosen programming language for the project is Java, selected for its advantageous characteristics, including high performance, extensive tool availability, cross-platform compatibility, rich libraries, cost-effectiveness (freely available), and a streamlined development process. Java's versatility enhances the app's capability to handle real-time weather data efficiently, providing users with a robust and reliable tool for staying informed about current and upcoming weather conditions.

1.3 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 Specificatio

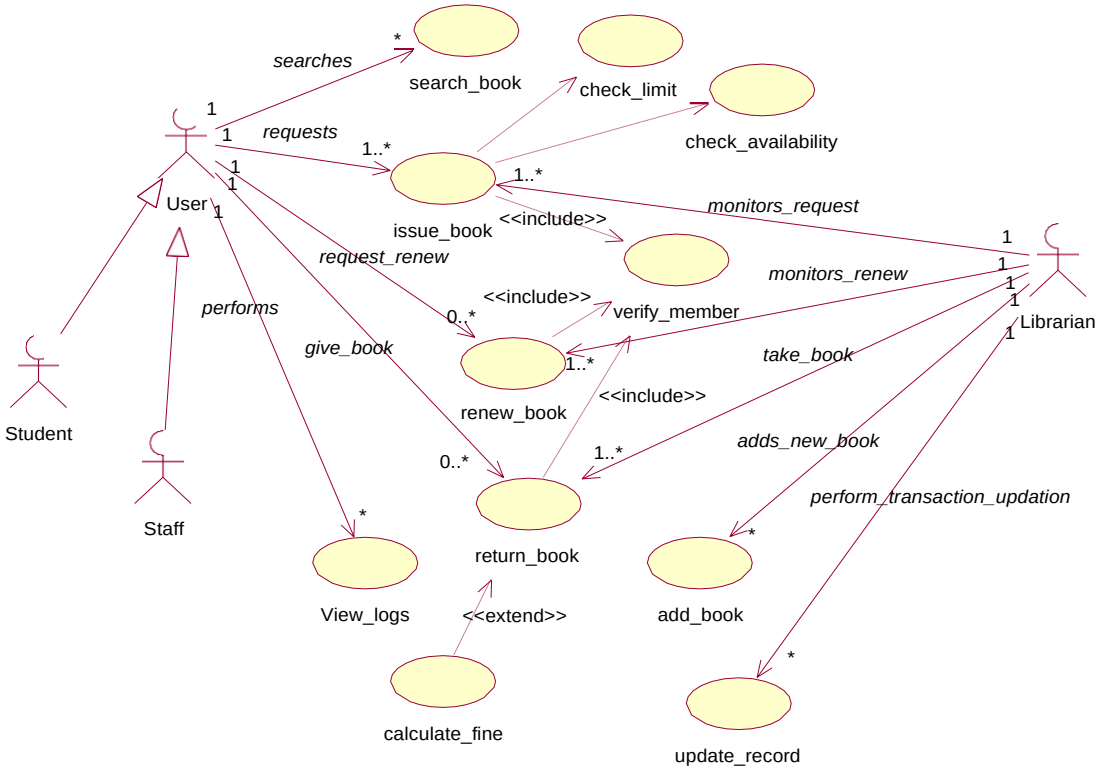
1.4 References

- 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://ebookily.net/doc/srs-library-management-system**

2. Overall Descriptions

2.1 Product Perspective

Use Case Diagram of Library Management System

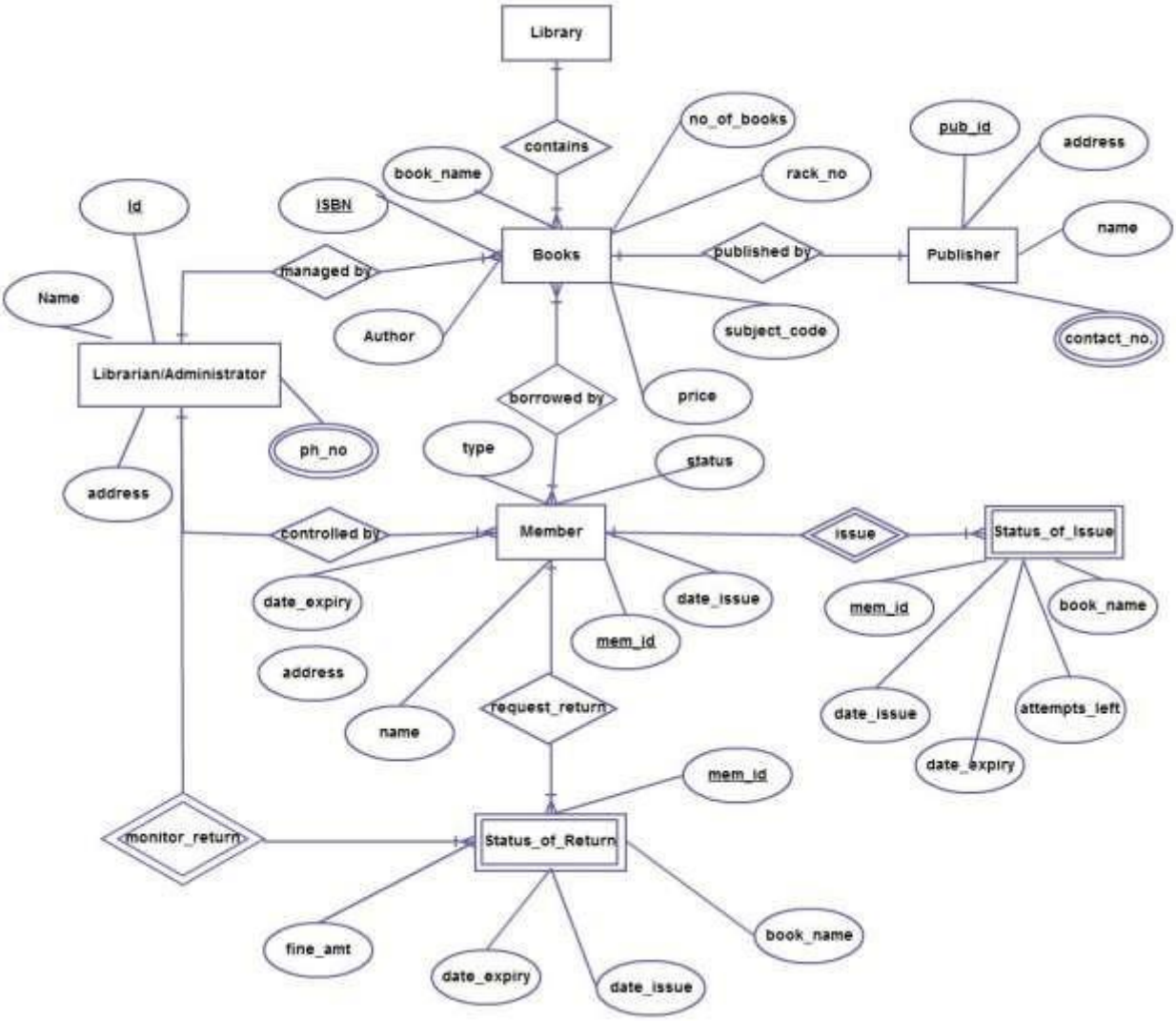


This is a broad level diagram of the project showing a basic overview. The users can be either staff or student.. This System will provide a search functionality to facilitate the search of resources. This search will be based on various categories viz. book name or the ISBN. Further the library staff personnel can add/update the resources and the resource users from the

system.The users of the system can request issue/renew/return of books for which they would have to follow certain criteria.

2.2 Product Function

Entity Relationship Diagram of Library Management System



The Online Library System provides online real time information about the books available in the Library and the user information. The main purpose of this project is to reduce the manual work. This software is capable of managing Book Issues, Returns, Calculating/Managing Fine, Generating various Reports for Record-Keeping according to end user requirements. The Librarian will act as the administrator to control members and manage books. The member's status of issue/return is maintained in the library database. The member's details can be fetched by the librarian from the database as and when required. The valid members are also allowed to view their account information.

2.3 User Classes and Characteristics

User to City:

Verb: Searches For
Cardinality: One user searches for one or more cities, and each city can be searched by multiple users.
User to Date:

Verb: Makes Request On
Cardinality: One user makes a request on multiple dates, and each date can be requested by multiple users.
City to Weather:

Verb: Has Weather Data
Cardinality: Each city has weather data for multiple dates, but each weather data corresponds to one particular city.
Date to Weather:

Verb: Corresponds To
Cardinality: Each date corresponds to one set of weather data, and each set of weather data corresponds to one date.

2.4 Operating Environment

The Weather App With Database is designed to operate seamlessly across various mobile platforms. Its primary target environments are Android and iOS devices, leveraging the extensive reach and user base of these ecosystems. To achieve optimal performance and user experience, the app will be developed with native libraries and frameworks specific to each platform.
While the primary focus lies on mobile platforms, the project also considers the potential for future expansion. The modular architecture allows for the development of web-based interfaces, enabling access on desktops and laptops. The basic input devices required are keyboard, mouse and output devices are monitor, printer etc.

2.5 Assumptions and Dependencies

The assumptions are:-

- The chosen DBMS is compatible with Java and supports efficient data storage and retrieval.
- Users enter accurate city names for retrieving relevant weather information.
- Users understand the displayed weather data and its units (metric or imperial).
- The user interface is intuitive and requires minimal guidance for users to navigate.

The dependencies are:-

- Smooth database operations are crucial for storing and retrieving weather data quickly.
- The app should function properly on various smartphone devices with different screen sizes and operating systems.
- Availability of compatible development tools and libraries for Java and the chosen Dbms is crucial for building the app.
- Secure data storage and transmission are essential to protect user privacy and prevent unauthorized access to sensitive information.

2.6 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/macOS/Linux (depending on developer preference and chosen tools)

Language: Java Development Kit (JDK) version 11 or later

Database: MS SQL Server (back end)

Hardware Configuration:-

Developer machine with sufficient processing power, RAM, and storage

Processor: Pentium(R)Dual-core CPU

Hard Disk: 40GB

RAM: 256 MB or more

2.7 Data Requirement

The Weather App with Database necessitates data input for user location selection within the application. Users initiate queries, seeking information on temperature, humidity, pressure, visibility, wind speed, dew point, UV index, moon phase, and air quality. The application, developed in Java and integrated with a Database Management System (DBMS), processes these queries and retrieves weather data from the database, tailored to the selected location for a Particular week. The database comprehensively stores weather details for each location, ensuring historical and current data accessibility. User account data, including chosen locations, is securely stored within the DBMS. The application boasts a responsive Java-based interface, providing users with seamless access to the requested weather data stored in the database.

3. External Interface Requirement

3.1 GUI

The Weather App With Database will present users with a visually appealing and intuitive graphical interface. The design adheres to standard principles, ensuring an enhanced user experience and easy navigation.

User Interface Design Principles: The Weather App With Database will present users with a visually appealing and intuitive graphical interface. The design adheres to standard principles, ensuring an enhanced user experience and easy navigation.

Customization: Users will have the flexibility to customize the interface according to their preferences.

Modules Integration: All modules within the software, such as user management, weather information display, and date selection, seamlessly integrate into the graphical user interface.

Login Interface:

Users are required to log in using a username and password. New users can register by entering their details, and incorrect login attempts trigger error messages.

Search Feature:

Users can search for weather information by entering the name of the city and date.

Categories View:

The app displays categories of weather information, including Temperature, Humidity, Pressure, Visibility, Wind, DewPoint, UV Index, Moon Phase, and Air Quality..

4. System Features

The Weather App With Database focuses on user account security and delivering weather updates with the following features:

- User Authentication and Validation: Users securely access the app with unique credentials and member ID verification, ensuring controlled and verified access.
- Monitoring and Weather Alerts: The system actively monitors user activity and provides real-time weather alerts. Users receive pop-up notifications for imminent weather changes or events, such as rain, allowing them to stay informed about the current conditions.
- Real-time Weather Display: The app prioritizes real-time weather data, offering users the latest and most accurate information about current weather conditions.
- Accountability and Privacy: The system maintains proper accountability by limiting access to individual accounts. Users can securely view and manage their real-time weather information, ensuring privacy and data security. The absence of an administrator role contributes to a straightforward and secure user experience.

5. Other Non-functional Requirements

5.1 Performance Requirement

The Weather App demands a user-centric performance experience.

- Performance requirements may vary based on target user base and device capabilities.
- Monitoring and logging are vital for identifying and addressing performance issues before they impact users.
- Offline functionality is crucial, with cached data displaying even without internet.
- Network bandwidth needs to be used efficiently, especially for low-bandwidth environments like mobile data. The app should scale seamlessly with more users and data without performance hiccups.

5.2 Safety Requirement

The Weather App primarily focuses on information and user convenience, safety considerations are still crucial to ensure user well-being and prevent potential harm.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.

5.3 Security Requirement

- Secure user login with username/password or other strong authentication methods (e.g., multi-factor authentication).
- Validate Database responses to prevent data tampering and injection attacks.
- Implement a secure development lifecycle (SDLC) with regular security testing and vulnerability assessments.
- Securely handle user location data if collected, with clear privacy policies and user consent

5.4 Requirement attributes

- All necessary features and functionalities are defined to fulfill user needs and project goals.
- Availability and uptime requirements for the app and database, ensuring consistent user access.
- The assumptions and dependencies that are mentioned above need to be addressed for successful implementation.

5.5 Business Rules

5.6 Weather data flows in every 3 hours, stored for your location and vicinity, neatly organized in current, past, and future segments. Searching by city or GPS feels effortless, with suggestions dancing to your fingertips. Performance should be good, with fast loading and data retrieval, fueled by caching and optimization. Scalability stretches its wings, ready to embrace a growing user base. These business rules are the backbone of your app, ensuring accuracy, privacy, user-friendliness, and a weather experience that truly delivers.

5.7 User Requirement

The below user requirements represent the core expectations and desires of your target audience. Meeting these needs will ensure your Weather App with Database delivers a valuable, user-centric experience that keeps users coming back for more. Remember, user feedback is invaluable, so listen closely and continue adapting and evolving your app to exceed their expectations.

The admin provides certain facilities to the users in the form of:-

- Backup and Recovery
- Accurate weather information for their current location and surrounding area.
- Users want to personalize their experience by setting default units, data format, and notification preferences.
- Users expect their data (location, search history) to be stored securely and used responsibly.
- Users want to personalize their experience by setting default units, data format, and notification preferences.
- The server must be maintained regularly and it has to be updated from time to time

6. Other Requirements

6.1 Data and Category Requirement

There are different categories namely Weather data, including temperature, humidity, pressure, and more, needs to be stored and accessed efficiently. Location data, including city names, coordinates, and timestamps, is crucial for providing localized weather information. User preferences, such as default units and notification settings, cater to individual needs. A relational database, like MySQL or PostgreSQL, ensures organized data storage and retrieval. Separate tables for different data categories maintain data integrity. Foreign key relationships bind tables together, preventing data inconsistencies. The categories and the data related to each category should be coded in the particular format.

6.2 Appendix

A: Admin, Abbreviation, Acronym, Assumptions; B: Books, Business rules; C: Class, Client, Conventions; D: Data requirement, Dependencies; G: GUI; K: Key; L: Library, Librarian; 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;

6.3 Glossary

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

- Administrator: A login id representing a user with user administration privileges to the software
- 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

6.4 Class Diagram

A class is an abstract, user-defined description of a type of data. It identifies the attributes of the data and the operations that can be performed on instances (i.e. objects) of the data. A class of data has a name, a set of attributes that describes its characteristics, and a set of operations that can be performed on the objects of that class. The classes' structure and their relationships to each other frozen in time represent the static model. In this project there are certain main classes

which are related to other classes required for their working. There are different kinds of relationships between the classes as shown in the diagram like normal association, aggregation, and generalization. The relationships are depicted using a role name and multiplicities. Here ‘User’, ‘City’,’Date’ and ‘Weather’ are the most important classes which are related to other classes.