Weather App with DB

Prepared by:-

Algotutor Academy

1. Introduction

1.1 Purpose

The main objective of this document is to outline the requirements for the project "Weather App with Database Integration (DB)." The document provides a comprehensive description of both functional and non-functional requirements as specified by the client. The primary goal of this project is to create a user-friendly application that utilizes a database for storing and retrieving weather-related data efficiently.

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

1.3 Scope of Development Project

The primary objective of this project is to develop a "Weather App with Database Integration (DB)" that serves as an intuitive and user-friendly platform for accessing real-time and historical weather information. This application aims to streamline the retrieval of weather data by incorporating a reliable database, enabling users to receive accurate updates based on their location or preferences. Functional requirements include the seamless integration of a database to store weather-related data, user profiles, and other relevant information. The system should retrieve real-time weather data from external APIs and efficiently store it in the database. User authentication features are essential to ensure that only authorized individuals can access both current and historical weather information. Location-based services must allow users to receive weather updates based on their current location or any designated area of interest. The application should also employ forecasting capabilities by analyzing historical data stored in the database. A user-friendly interface is crucial, facilitating easy navigation and interaction with both real-time and historical weather information. Additionally, users should have the ability to set preferences for weather alerts through a notification system, with these preferences managed and stored in the database. Non-functional requirements emphasize the importance of performance, scalability, reliability, security, and compatibility. The application should efficiently retrieve and display weather information from the database, ensuring a responsive user experience. The database should be designed to handle the growing amount of weather data and user profiles. Reliability is paramount to ensure accurate weather information retrieval and seamless database operations. Robust security measures must be implemented to safeguard user data and maintain the integrity of the weather database. The application should be compatible with various devices and operating systems, while comprehensive documentation, including ER diagrams and UML diagrams, will aid in understanding system architecture and interactions. In summary, similar to the Library Management System described earlier, the Weather App with Database Integration aims to modernize and enhance the user experience by utilizing a database to store and manage weather-related information efficiently. The choice of Java as the development language ensures advantages in terms of performance, cross-platform compatibility, and development tools.

1.4 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

1.5 References

* Books
* Book Title: "Database Systems: Design, Implementation, and Management" Author: Carlos Coronel, Steven Morris
* Book Title: "The AMS Weather Book: The Ultimate Guide to America's Weather" Author: Jack Williams
* Book Title: "Effective Java" Author: Joshua Bloch
* Websites
* W3Schools - SQL Tutorial
* Microsoft Learn - Database Fundamentals

2. Overall Descriptions

2.1 Product Perspective

Use Case Diagram of Weather App with Database Integration

This high-level diagram provides an overview of the Weather App with Database Integration, illustrating the key functionalities and interactions. Users of the system can be categorized into regular users and administrators. The system's primary function is to provide real-time and historical weather information based on user preferences and location.

sql

Copy code

+------------------------------+

| Weather Application |

+------------------------------+

| |

| |

| |

| User |

| \* |

| 1..\* |

| |

| view\_weather |

| |

| set\_preferences |

| |

| receive\_alerts |

| |

| |

+------------------------------+

|

|

+-----------------------------+

| Administrator |

+------------------------------+

| |

| |

| \* |

| 1..\* |

| |

| manage\_data |

| |

| view\_reports |

| |

+---------------------------+

Use Case Descriptions:

User:

View Weather: Users can view real-time weather information based on their preferences and location.

Set Preferences: Users can set their preferences for weather alerts and data presentation.

Receive Alerts: Users can receive weather alerts based on their set preferences.

Administrator:

Manage Data: Administrators can manage weather data, ensuring its accuracy and reliability in the database.

View Reports: Administrators can view reports related to weather patterns, user interactions, and system performance.

This use case diagram highlights the key interactions between users and the weather application. The system utilizes a database to store and manage weather data efficiently, enabling users to access personalized weather information and administrators to maintain the integrity of the system. The diagram includes functionalities such as viewing weather, setting preferences, receiving alerts, managing data, and viewing reports.

2.2 Product Function

The Entity-Relationship Diagram (ERD) for the Weather App with Database Integration provides a comprehensive visual representation of the structured relationships among key entities within the system. This application has been developed to deliver real-time weather information, leveraging a robust database for streamlined data storage and retrieval. The overarching objective is to optimize access to weather data while minimizing manual interventions. At the core of the system are distinct entities, including "User," "Weather Data," "Administrator," "WeatherAlerts," and "UserPreferences." Users, identified uniquely by their UserIDs, embody individuals engaging with the weather app. The User entity captures pertinent details such as Name, Address, Email, and user-specific preferences like notification settings. On the other hand, the Weather Data entity encapsulates real-time and historical weather information, identified by a unique DataID, storing critical data points such as Temperature, Humidity, Wind Speed, and Atmospheric Pressure. Administrators, denoted by AdminID, play a pivotal role in managing weather data and system settings. The Administrator entity encompasses key details such as the administrator's Name, Username, and Password. The system's responsiveness to dynamic weather conditions is highlighted by the WeatherAlerts entity, which records alerts with a unique AlertID, intricately linked to relevant Weather Data through DataID. These alerts include information such as AlertType, AlertMessage, and Timestamp. Furthermore, the UserPreferences entity manages individual user settings, identified by a unique PreferenceID and linked to UserID for personalized configurations. Preferences encompass choices related to NotificationType, Location, and AlertThresholds, offering users a tailored experience. The ERD visually articulates the interconnections and dependencies between users, weather data, administrators, weather alerts, and user preferences. Through seamless data retrieval and user interactions, the database efficiently manages the intricate relationships within the system. This design underscores the significance of structured relationships and effective database integration, aligning cohesively with the overarching goal of providing users with an intuitive, user-friendly, and efficient platform for weather management.

2.3 User Classes and Characteristics

The system provides distinct services tailored to the specific roles of its users, categorized as either "Administrator" or "User" in the context of the Weather App with Database Integration. The Administrator, similar to a Librarian in the Library Management System, assumes a controlling role with comprehensive privileges. Users, in this case, include both general users and administrators who oversee weather data management.

Administrator Features:

* Manage Weather Data: Administrators can oversee and manipulate real-time and historical weather data within the application.
* View Weather Categories: Administrators have the capability to explore various weather categories available in the system.
* Access Detailed Weather Lists: They can peruse detailed lists of weather data within each category.
* Data Maintenance: Administrators can handle the reception and incorporation of weather updates.
* Update Database: They have the authority to add new weather information and edit details of existing data entries.
* Generate Reports: Administrators can generate reports summarizing the current weather data and historical trends.
* User Account Access: They can access and manage user accounts, ensuring the integrity of user information.

User Features:

* Explore Weather Categories: Users can navigate and explore different weather categories within the application.
* Access Detailed Weather Lists: They have the ability to view detailed lists of weather data corresponding to each category.
* Create and Manage User Account: Users can establish and maintain their personal accounts within the weather app database.
* View Personal Weather Data: Users can check and keep track of weather information specific to their preferences and locations.
* Request New Weather Data: They can submit requests for specific weather data not currently available.
* Review Weather History: Users have the capability to review their historical interactions with weather data.
* Search for Specific Weather Information: Users can perform targeted searches to find particular weather information of interest.

In this adaptation, the features for the Weather App align with the objectives of the Library Management System but are tailored to the context of weather data. Administrators possess comprehensive control over weather data management and user accounts, while general users enjoy a range of features facilitating personalized interactions with weather information.

2.4 Operating Environment

The Weather App with Database Integration is designed to operate seamlessly within the Windows environment, offering compatibility across widely used browsers such as Microsoft Internet Explorer, Google Chrome, and Mozilla Firefox. For the purposes of the model, the application ensures optimal functionality on Microsoft Internet Explorer 6.0 and is largely compatible with Mozilla Firefox and Opera 7.0 or higher versions. The user-friendly interface caters to diverse browser preferences, ensuring a consistent experience across platforms. The sole prerequisite for utilizing this online weather application is a stable internet connection.

2.5 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 information of all users, books and libraries must be stored in a database that is accessible by the website
* The system should have more storage capacity and provide fast access to the database
* The system should provide search facility and support quick transactions
* The Library System is running 24 hours a day
* 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

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 (admin) should have proper understanding of the product
* The system should have the general report stored
* The information of all the users must be stored in a database that is accessible by the Library System
* Any update regarding the book from the library is to be recorded to the database and the data entered should be correct

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 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

2.7 Data Requirement

The inputs consist of the query to the database and the output consists of the solutions for

the query. The output also includes the user receiving the details of their accounts. In this project

the inputs will be the queries as fired by the users like create an account, selecting books and

putting into account. Now the output will be visible when the user requests the server to get

details of their account in the form of time, date and which books are currently in the account.

3. External Interface Requirement

3.1 GUI

The software provides good graphical interface for the user and the administrator can operate on

the system, performing the required task such as create, update, viewing the details of the book.

* It allows user to view quick reports like Book Issued/Returned in between particular time.
* It provides stock verification and search facility based on different criteria.
* The user interface must be customizable by the administrator
* All the modules provided with the software must fit into this graphical user interface and accomplish to the standard defined
* The design should be simple and all the different interfaces should follow a standard template
* The user interface should be able to interact with the user management module and a part of the interface must be dedicated to the login/logout module

Login Interface:-

In case the user is not yet registered, he can enter the details and register to create his account.

Once his account is created he can ‘Login’ which asks the user to type his username and

password. If the user entered either his username or password incorrectly then an error message

appears.

Search:-

The member or librarian can enter the type of book he is looking for and the title he is interested

in, then he can search for the required book by entering the book name.

Categories View:-

Categories view shows the categories of books available and provides ability to the librarian to

add/edit or delete category from the list.

Librarian’s Control Panel:-

This control panel will allow librarian to add/remove users; add, edit, or remove a resource. And

manage lending options.

4. System Features

The users of the system should be provided the surety that their account is secure. This is

possible by providing:-

* User authentication and validation of members using their unique member ID
* Proper monitoring by the administrator which includes updating account status, showing a popup if the member attempts to issue number of books that exceed the limit provided by the library policy, assigning fine to members who skip the date of return
* Proper accountability which includes not allowing a member to see other member’s account. Only administrator will see and manage all member accounts

5. Other Non-functional Requirements

5.1 Performance Requirement

The proposed system that we are going to develop will be used as the Chief performance system

within the different campuses of the university which interacts with the university staff and

students. Therefore, it is expected that the database would perform functionally all the

requirements that are specified by the university.

* The performance of the system should be fast and accurate
* Library Management System shall handle expected and non-expected errors in ways that prevent loss in information and long downtime period. Thus it should have inbuilt error testing to identify invalid username/password
* The system should be able to handle large amount of data. Thus it should accommodate high number of books and users without any fault

5.2 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.

5.3 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.

5.4 Requirement attributes

* There may be multiple admins creating the project, all of them will have the right to create changes to the system. But the members or other users cannot do changes
* The project should be open source
* The Quality of the database is maintained in such a way so that it can be very user friendly to all the users of the database
* The user be able to easily download and install the system

5.5 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.

5.6 User Requirement

The users of the system are members and Librarian of the university who act as administrator to

maintain the system. The members are assumed to have basic knowledge of the computers and

internet browsing. The administrators of the system should have more knowledge of the internals

of the system and is able to rectify the small problems that may arise due to disk crashes, power

failures and other catastrophes to maintain the system. The proper user interface, user manual,

online help and the guide to install and maintain the system must be sufficient to educate the

users on how to use the system without any problems.

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

* Backup and Recovery
* Forgot Password
* Data migration i.e. whenever user registers for the first time then the data is stored in the server
* Data replication i.e. if the data is lost in one branch, it is still stored with the server
* Auto Recovery i.e. frequently auto saving the information
* Maintaining files i.e. File Organization
* 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

The Weather App with Database Integration caters to various user categories, including general users, administrators, and specific user roles such as meteorologists. Access rights are determined based on the user's category; administrators have comprehensive rights to modify, delete, and append data, while other users, except meteorologists, have rights limited to retrieving information from the database. The application also classifies weather data into different categories. Depending on the type of weather information sought, the relevant data should be displayed. These categories and their associated data should adhere to a standardized format.

6.2 Appendix

A: Admin, Abbreviation, Acronym, Assumptions; B: Weather Data, Business rules; C: Class, Client, Conventions; D: Data requirement, Dependencies; G: GUI; K: Key; L: Librarian, Location-based Data; M: Meteorologist; 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 glossary outlines conventions and acronyms used in this document and the project:

* Administrator: A login ID representing a user with administration privileges for the weather app.
* 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: Refers to what the user interacts with directly.
* Application Logic Layer: Refers to the Web Server, where computations are completed.
* Data Storage Layer: Refers to where all weather data is recorded.
* Use Case: A broad-level diagram showing a basic overview of the project.
* Class Diagram: A static structure diagram describing the structure of the system, including classes, attributes, and relationships.
* Interface: Something used to communicate across different mediums.
* Unique Key: Used to differentiate entries in a database.

6.4 Class Diagram

In the Weather App with Database Integration, a class represents an abstract, user-defined description of a type of weather data. It identifies attributes and operations that can be performed on instances of the data. The static model captures the structure and relationships between classes. Important classes, such as 'Meteorologist,' 'User,' and 'Weather Data,' are interconnected with other classes. Various relationships, including normal association, aggregation, and generalization, are depicted in the diagram, providing a comprehensive overview of the system's class structure and interactions.