
Software Requirements Specification

for

<Personal Finance Tracker>

Version <1.1>

Prepared by

Group Name: Finance Frontiers

Abu Bakr Majid
Zain Allaudin
Mishaal Rafique
Eshal Shahid
Ahsan Ali

23L-3050
23L-3036
23L-3044
23L-3086
23L-3031

l233050@lhr.nu.edu.pk
l233036@lhr.nu.edu.pk
l233044@lhr.nu.edu.pk
l233086@lhr.nu.edu.pk
l233031@lhr.nu.edu.pk

Instructor: *Ms. Kiran Khurshid*

Course: *Introduction to Software Engineering*

Project Domain: *Finance*

Teaching Assistant: *Ms. Areeba Mujahid*

Date: *01st April, 2024*

Table of Contents

Table of Contents	2
1 Introduction	1
1.1 Product Scope	1
2 Overall Description	2
2.1 Product Overview	2
2.2 Product Functionality	2
2.3 Assumptions and Dependencies	2
3 Specific Requirements	3
3.1 External Interface Requirements (Will be made in the design phase of your project)	3
3.2 Functional Requirements	3
3.3 Use Case Model	3
4 Other Non-functional Requirements	5
4.1 Performance/Safety and Security Requirements/Software Quality Attributes	5

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Zain Allaudin Mishaal Rafique Abu Bakr Majid Eshal Shahid Ahsan Ali	The first version of the Software Requirements Specification (SRS) for the Personal Finance Tracker aims to outline the essential functionalities and system requirements necessary for the development of the application.	14/03/24
1.1	Zain Allaudin Mishaal Rafique Abu Bakr Majid Eshal Shahid Ahsan Ali	In second version of the SRS, the user interface of Personal Finance Tracker is described concisely, focusing on its layout, design elements, and user interaction.	01/04/2024

1. Introduction

1.1 Product Scope

The Personal Finance Tracker, is an ambitious software project aimed at providing an efficient way to the individuals to manage their finances. This sophisticated tool is designed to simplify and enhance financial management practices. The primary purpose of the software is to empower users with comprehensive insights into their financial health, enabling informed decision-making and goal achievement.

With features like user authentication, multi-currency and account support, expense tracking, and budgeting tools, the Personal Finance Tracker offers a holistic approach to financial management. Users can securely manage their finances, track expenses across various accounts and currencies, and set and monitor progress towards financial goals. By addressing common challenges such as data security, intuitive interface design, and multi-currency transactions, the software aims to deliver a user-friendly, secure, and effective financial management tool. Ultimately, the Personal Finance Tracker seeks to fill gaps in current personal finance management tools by offering a tailored, all-encompassing solution that enhances users' financial literacy and control over their monetary resources.

2. Overall Description

2.1 Product Overview

The Personal Finance Tracker is a standalone software solution designed to address the need for efficient and user-friendly financial management tools. It operates within the user's personal financial landscape, providing features such as expense tracking, budgeting tools, and goal setting. The software interacts directly with the user and interfaces with external systems or services as required, facilitating tasks like currency conversion.

2.2 Product Functionality

- *User Authentication*
- *Tracking Income and Expenses*
- *Multi-Currency Support*
- *Multi-Account Support*
- *Categorizing Spendings and Expenses*
- *Budget Calculation*
- *Budget Goals*
- *Transaction History*
- *Budget Analysis*
- *Powerful Reporting and Visualization*
- *Flexible Transaction Management*
- *Data Security*

2.3 Assumptions and Dependencies

Assumptions:

1. **User Device Compatibility:** Users are assumed to use devices compatible with the application's platform (e.g., Windows OS), ensuring optimal performance.
2. **User Financial Literacy:** Users are assumed to possess basic financial literacy for effective utilization of the application's features.

Dependencies:

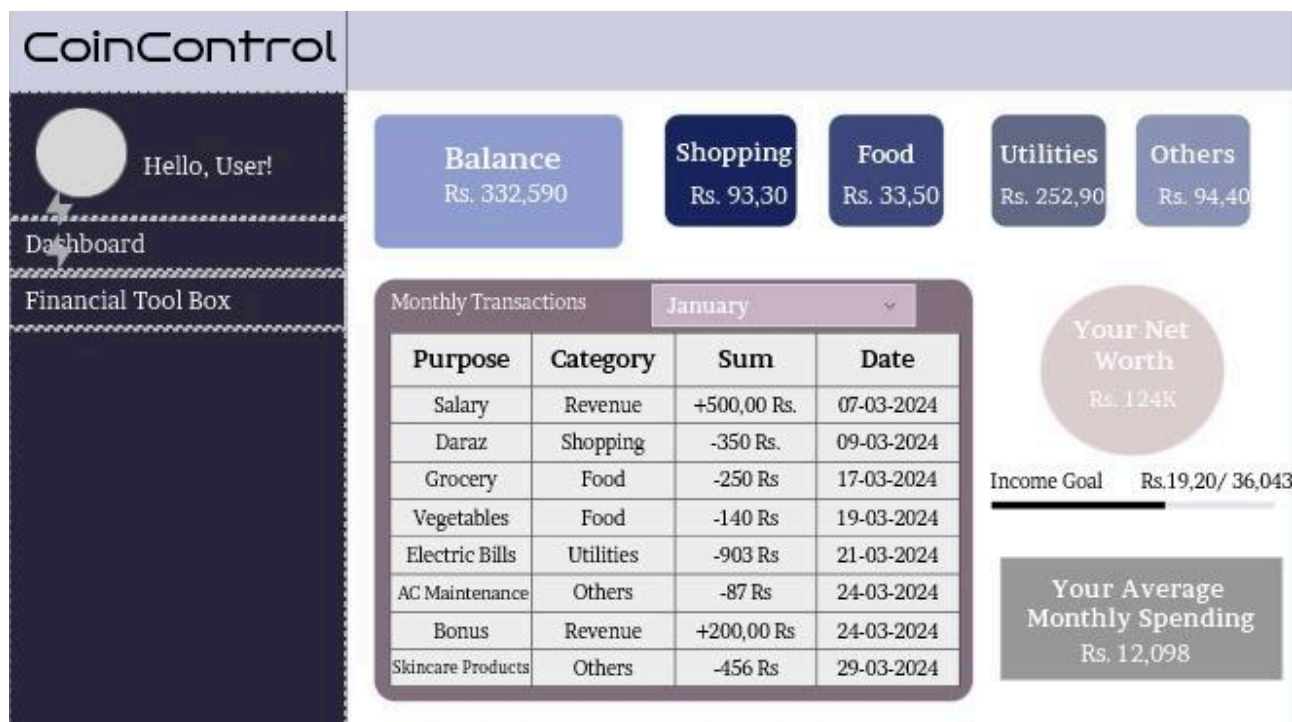
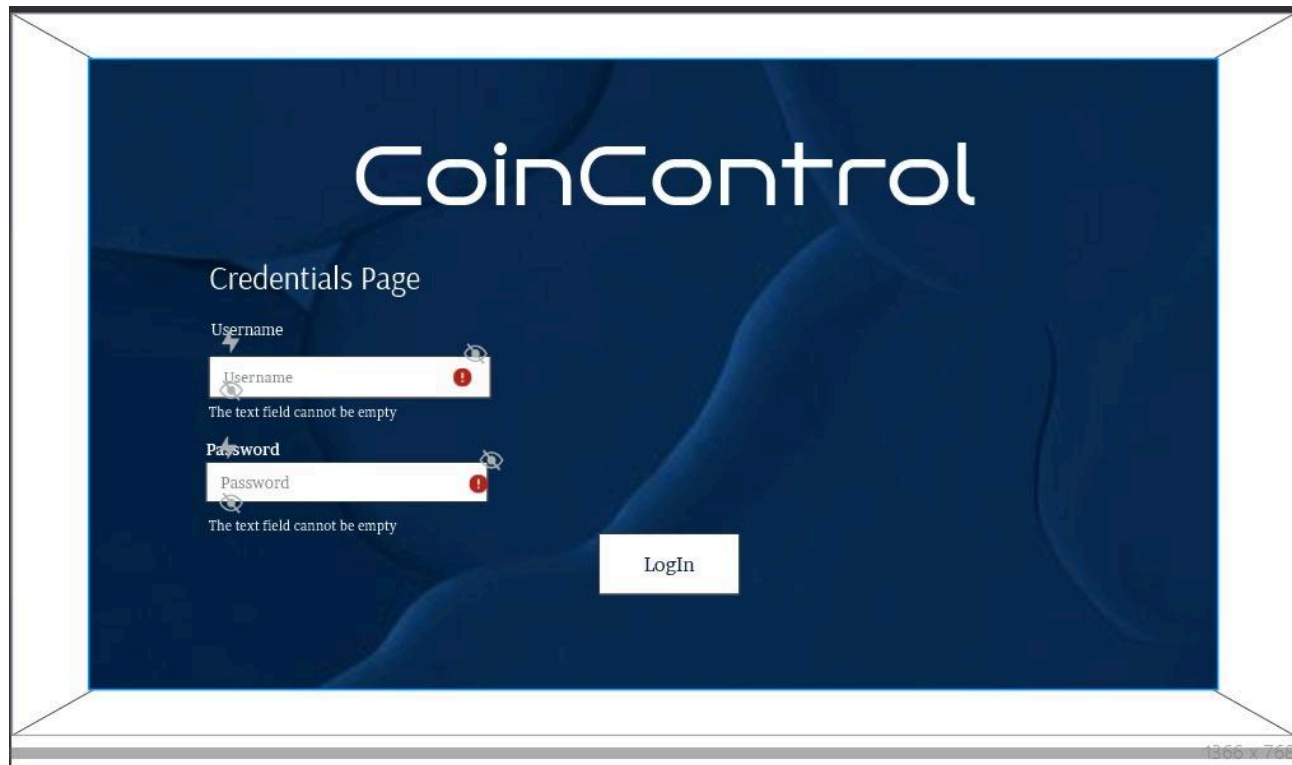
1. **Database Management System (DBMS):** The project relies on a suitable DBMS (e.g., MySQL, SQLite) for storing and managing user data.

3. Specific Requirements


3.1 External Interface Requirements

3.1.1 User Interfaces

1. Purpose and Components:
 - The user interface (UI) for a personal finance tracker serves as the bridge between users and the underlying software.
 - Components that require a user interface include:
 - **Data Input:** Users need to input financial transactions (e.g., income, expenses).
 - **Data Display:** Users should be able to view summary reports.
 - **Data Manipulation:** Users may want to analyze data, set budgets, and plan for financial goals.
2. User-Friendly Design:
 - **Ease of Learning:** A good UI should be easy to learn. Avoid complex syntax and semantics for command input.
 - **Metaphors and Intuitive Commands:** Use metaphors (e.g., financial tool box) and intuitive command names to enhance learning speed.
3. Reporting and Planning:
 - **Essential Reports:** Generate standard financial reports (monthly expenditure sheet).
4. Visual Representation:
 - **Data Visualization:** Use dashboards to help users understand their financial data.
 - **Stunning UI:** A clear and accurate UI enhances user experience and decision-making.
5. Use of Buttons:
 - **Standard Buttons:** Several standard buttons and functions are consistently available across all screens to ensure access to essential tools for managing personal finances.



CoinControl

Hello, User!

[Dashboard](#)

[Financial Tool Box](#)

Add Cash

Purpose

Sum

04/15/2024

Submit

Add Expenditure

Purpose

Sum

04/01/2024

Category

☐ Shopping ☒ Food ☐ Utilities ☐ Others

Submit

Goal Setter

Select Month

January

Goal Amount

Submit

Change Currency

Currency

US Dollar

Submit

Successfully Changed Currency!

3.2 Functional Requirements

3.2.1 F1: The system shall provide user authentication functionality, allowing users to securely create accounts and log in through individual verification.

3.2.2 F2: The system shall enable users to track their income and expenses by effortlessly recording transactions with details such as amount, category, and date.

3.2.3 F3: The system shall support multi-currency functionality, allowing users to handle transactions in different currencies like dollars, euros, and rupees.

3.2.4 F4: The system shall offer multi-account support, enabling users to track balances and transactions across various accounts, providing a comprehensive fiscal overview.

3.2.5 F5: The system shall allow users to categorize spendings and expenses, organizing transactions into various groups for effortless budgeting and analysis.

3.2.6 F6: The system shall calculate budgets by deducting expenses from income, providing users with an overview of their financial situation.

3.2.7 F7: The system shall enable users to set monthly spending and saving goals, track progress, and achievements to help them manage their finances effectively.

3.2.8 F8: The system shall maintain a transaction history, recording all transactions with dates, categories, and amounts for reference and analysis.

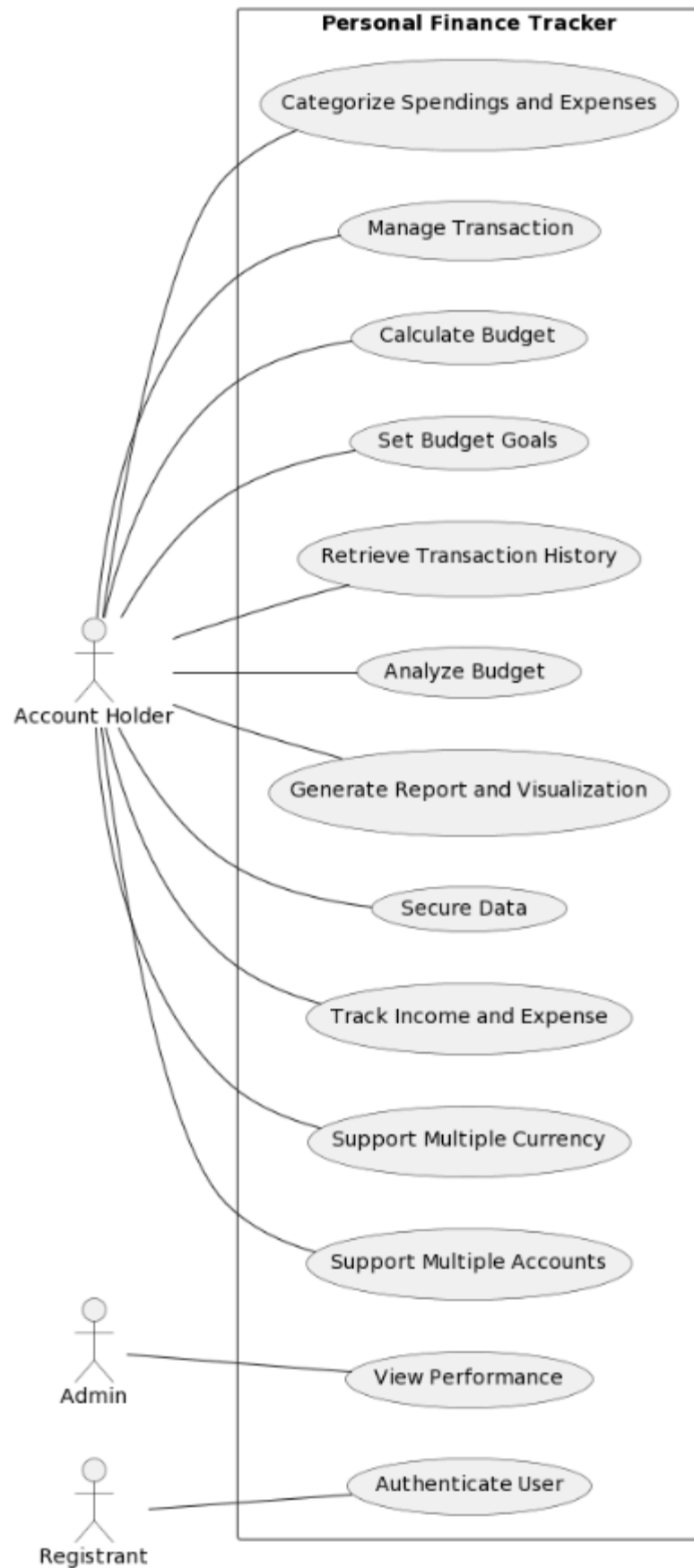
3.2.9 F9: The system shall provide budget analysis functionality, offering detailed insights into spending patterns, savings, and adherence to budget goals.

3.2.10 F10: The system shall offer reporting and visualization tools, generating insightful reports to help users understand their spending trends and identify areas for improvement.

3.2.11 F11: The system shall facilitate flexible transaction management, allowing users to add, and delete transactions with ease, categorizing them for in-depth analysis.

3.3 Use Case Model

USE CASE DIAGRAM



3.3.1 Calculate Budget (UC-1)

Author – Zain Allaudin

Purpose – Enables users to calculate their budget by deducting expenses from income.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User selects the budget calculation option.
2. System retrieves the user's income and expenses data.
3. System calculates the budget by deducting total expenses from total income.
4. System displays the calculated budget to the user.

Post conditions – Budget has been calculated and displayed to the user

3.3.2 Set Budget Goals (UC-2)

Author – Zain Allaudin

Purpose – Allows users to set monthly spending and saving goals, track progress, and achievements.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User navigates to the budget goals section.
2. Users set monthly spending and saving goals.
3. System saves the goals and tracks the user's progress.

Post conditions – Goals have been set, and progress has been tracked.

3.3.3 Retrieve Transaction History (UC-3)

Author – Zain Allaudin

Purpose – Enables users to maintain a record of all transactions with dates, categories, and amounts.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User accesses the transaction history section.
2. System retrieves and displays the user's transaction history.

Post conditions – Transaction history has been displayed to the user.

3.3.4 Analyze Budget (UC-4)

Author – Eshal Shahid

Purpose – Provides a detailed analysis of spending patterns, savings, and adherence to budget.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User navigates to the budget analysis section.
2. System retrieves and analyzes the user's spending patterns and savings.
3. System generates a detailed analysis report.
4. System displays the analysis report to the user.

Post conditions – Analysis report has been generated and displayed to the user.

3.3.5 Generate Report and Visualization (UC-5)

Author – Eshal Shahid

Purpose – Generates insightful reports to understand spending trends and identify areas for improvement.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User selects the reporting and visualization option.
2. System retrieves the user's financial data.
3. System generates insightful reports and visualizations based on the data.
4. System displays the reports and visualizations to the user.

Post conditions – Reports and visualizations have been generated and displayed to the user.

3.3.6 Secure Data (UC-6)

Author – Eshal Shahid

Purpose – Protects financial information with password protection and data encryption.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User accesses the data security settings.
2. User sets up a strong password for accessing the application.
3. System encrypts the user's financial data to protect it from unauthorized access.

Postconditions: User's financial data has been secured with password protection and encryption.

3.3.7 View Performance (UC-7)

Author – Mishaal Rafique

Purpose – Analyzes the performance of the application in handling transactions and generating reports.

Actors – Admin

Preconditions – None

Flow of Events –

1. System collects performance data related to transaction processing and report generation.
2. System analyzes the collected data to identify areas for performance improvement.

Postconditions – Performance analysis report has been generated for system optimization.

3.3.8 Authenticate User (UC-8)

Author – Mishaal Rafique

Purpose – Allows users to securely create accounts and log in.

Actors – Registrant

Preconditions – User is not authenticated.

Flow of Events –

1. Users enter their credentials (username and password) to create a new account.
2. System verifies the credentials and creates a new account for the user.
3. Users enter their credentials to log in to the application.
4. System verifies the credentials and authenticates the user.

Postconditions: User has been authenticated and can access the application.

3.3.9 Track Income and Expense (UC-9)

Author – Mishaal Rafique

Purpose – Allows users to record transactions with details like amount, category, and date.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User navigates to the income and expenses tracking section.
2. User records a new income or expense transaction with details like amount, category, and date.

Postconditions – Transaction has been recorded in the system.

3.3.10 Support Multiple Currency (UC-10)

Author – Abu Bakr Majid

Purpose – Allows users to handle transactions in different currencies.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User selects the currency for a new transaction.
2. System converts the transaction amount to the selected currency based on the current exchange rate.

Postconditions – Transaction amount has been converted and recorded in the specified currency.

3.3.11 Support Multiple Account (UC-11)

Author – Abu Bakr Majid

Purpose – Allows users to track balances and transactions across various accounts.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User selects the account for a new transaction.
2. System updates the account balance and transaction history for the selected account.

Postconditions – Account balance and transaction history has been updated.

3.3.12 Categorize Spendings and Expenses (UC-12)

Author – Ahsan Ali

Purpose – Allows users to organize transactions and categorize expenses into various groups.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User categorizes a new expense transaction into a predefined category.
2. System records the transaction under the specified category.

Postconditions – Transaction has been categorized for analysis.

3.3.13 Manage Transaction (UC-13)

Author – Ahsan Ali

Purpose – Allows users to add, edit, and delete transactions with ease, categorizing them for in-depth analysis.

Actors – Account Holder

Preconditions – User is authenticated.

Flow of Events –

1. User accesses the transaction management section.
2. User adds a new transaction with details like amount, category, and date.
3. User edits an existing transaction to update its details.
4. User deletes a transaction that is no longer needed.

Postconditions – Transactions have been managed as per user's actions.

4. Other Non-functional Requirements

4.1 Performance Requirements

4.1.1 Responsiveness

- The application must respond to user actions within 2 seconds under normal operating conditions.
- Response time for critical functionalities such as transaction recording and report generation should be less than 1 second.

4.1.2 Throughput

- The system shall be capable of handling a minimum of 1000 transactions per minute during peak usage periods.
- Throughput shall be scalable to accommodate increased user demand without degradation in performance.

4.2 Safety and Security Requirements

4.2.1 Data Encryption

- All sensitive user data, including financial transactions and personal information, must be encrypted using an encryption algorithm.
- Encryption should be applied both at rest and in transit to ensure data security throughout the application.

4.2.2 Access Control

- User authentication mechanisms should be robust to prevent unauthorized access to user accounts and data.

4.3 Software Quality Attributes

4.3.1 Adaptability:

- The software architecture should be designed to accommodate future changes and updates easily.
- Modularity and abstraction should be used to isolate components and minimize dependencies, allowing for flexible modifications.

4.3.2 Availability

- The system should aim for a minimum uptime over a specified time period, ensuring high availability for users.
- Redundancy mechanisms should be implemented to minimize downtime and ensure continuous service availability.

4.3.3 Correctness

- The application must ensure data accuracy by performing thorough validation checks on user input and transaction data.
- Automated data reconciliation processes should be in place to identify and resolve any discrepancies or errors in the system.

4.3.4 Flexibility

- Flexible configuration settings should allow users to adjust parameters such as budget categories and reporting intervals.

4.3.5 Maintainability

- The codebase should be well-structured and well-documented, following coding standards and best practices to facilitate ease of maintenance.
- Regular code reviews and refactoring should be conducted to keep the codebase clean and maintainable over time.

4.3.6 Portability

- The application should be designed to run seamlessly across desktop computers.

4.3.7 Reliability

- The system should be resilient to failures and errors, with built-in mechanisms to handle unexpected conditions gracefully.
- Error handling and recovery strategies should be implemented to minimize the impact of failures on system functionality and user experience.

4.3.8 Reusability

- The system should be built using modular components that can be reused in different parts of the application or in other projects.

4.3.9 Robustness

- The application must perform thorough input validation to prevent security vulnerabilities such as injection attacks and buffer overflows.
- Defensive programming techniques should be employed to handle unexpected inputs and edge cases effectively.

4.3.10 Testability

- The application should have comprehensive test coverage, including unit tests, integration tests, and end-to-end tests, to ensure thorough validation of system functionality.
- Test cases should be automated wherever possible to streamline the testing process and facilitate continuous integration and delivery.

4.3.11 Usability

- *The user interface should be intuitive and easy to navigate, with clear labeling, consistent layout, and minimal cognitive load.*
- *Usability testing should be conducted with real users to gather feedback and identify areas for improvement in the interface design.*