Software Requirements Specification

For

Automate Banking website

Version 1.0 approved

Prepared by Shejal Dhadse

HCL Training

Date: 10/01/2026

Submitted in partial fulfillment
of the requirements of
Software Engineering (Agile Methodology)

Abstract

The Automated Banking Website project aims to design and develop a secure, user-friendly, and efficient online banking platform using the Agile software development methodology. The primary objective of this project is to automate core banking services such as user registration, login authentication, account management, balance inquiry, fund transfer, bill payments, transaction history, and customer support, while ensuring high security, scalability, and reliability.

Agile methodology is adopted in this project to allow flexibility, continuous improvement, and faster delivery of functional modules. The project is divided into multiple short development cycles known as sprints. Each sprint focuses on developing, testing, and delivering specific features such as user authentication, account dashboard, transaction processing, and security enhancements. Regular stakeholder feedback is incorporated at the end of every sprint to ensure that the system meets user requirements and banking standards effectively. The Automated Banking Website reduces manual intervention, minimizes processing time, and improves overall customer satisfaction by providing 24×7 access to banking services.

Advanced security mechanisms such as role-based access control, encrypted transactions, and secure session management are integrated to protect sensitive financial data. The system is designed with responsiveness and usability in mind so that customers can access banking services through different devices. From a project management perspective, Agile practices such as daily stand-up meetings, sprint planning, sprint reviews, and retrospectives help in identifying risks early, improving team collaboration, and ensuring timely delivery of high-quality software. Continuous testing and integration ensure that defects are detected and resolved quickly.

In conclusion, the Automated Banking Website developed using Agile methodology provides a robust, secure, and scalable solution for modern banking needs. The use of Agile ensures adaptability to changing requirements, faster deployment, and continuous enhancement, making the system suitable for dynamic and customer-centric banking environments.

Table of Contents

List of Figures

List of Tables

2. Introduction

2.1 Introduction

The banking industry plays a vital role in the economic development of any country. With the advancement of information technology and widespread internet usage, the demand for online banking services has increased significantly. Customers expect fast, reliable, and secure banking services that can be accessed anytime and from anywhere. Automated banking systems fulfill these expectations by offering digital platforms that replace traditional manual banking operations.

An Automated Banking Website provides an integrated platform where customers can perform various banking activities without visiting a physical branch. These activities include checking account balance, transferring funds, paying utility bills, and reviewing transaction history. Automation improves efficiency, reduces paperwork, and enhances accuracy in financial operations. It also helps banks manage large volumes of transactions with minimal human intervention.

This project aims to develop an Automated Banking Website using Agile methodology. Agile supports iterative development and continuous feedback, making it suitable for complex and evolving systems like banking applications. The system is designed with a focus on security, usability, and scalability to meet modern banking requirements.

2.2 Problem Identification

Traditional banking systems face several challenges such as long queues, limited working hours, manual data entry, and high dependency on staff availability. Customers often experience delays in service delivery and inconvenience due to rigid banking processes. Manual handling of records increases the risk of errors, data inconsistency, and security breaches.

Another major issue is the lack of real-time access to banking information. Customers are unable to instantly view transaction updates or account details. These limitations highlight the need for an automated and online banking solution that offers speed, transparency, and reliability.

2.3 Need of the Project

The need for an Automated Banking Website arises from the growing demand for digital banking services. Automation reduces operational costs, improves service quality, and enhances customer satisfaction. It enables banks to provide uninterrupted services and improves data accuracy and security.

Key needs of the project include:

- 24×7 availability of banking services
- Reduction in manual workload
- Improved transaction speed and accuracy
- Secure handling of customer data
- Enhanced customer experience

2.4 Project Scheduling

The project follows Agile-based scheduling, where development is divided into multiple sprints. Each sprint focuses on specific functionalities and delivers a working module.

Sprint	Duration	Activities
Sprint 1	2 Weeks	Requirement Analysis, Planning
Sprint 2	2 Weeks	UI Design, Login & Registration
Sprint 3	2 Weeks	Account Management, Transactions
Sprint 4	2 Weeks	Testing, Review, Deployment

This scheduling approach ensures timely delivery and continuous improvement.

2.5 Objectives

The main objectives of the Automated Banking Website project are:

- To automate core banking operations
- To provide a secure and reliable online banking system
- To improve customer convenience and satisfaction
- To reduce operational costs and manual errors
- To implement Agile development practices effectively

3. Software Requirement Specification (SRS)

3.1 Purpose

The purpose of the Software Requirement Specification (SRS) is to clearly define the functional and non-functional requirements of the Automated Banking Website. This document serves as a formal agreement between stakeholders, developers, and testers, ensuring that all system requirements are clearly understood before development begins. The SRS helps in reducing ambiguity, minimizing development errors, and providing a strong foundation for system design, implementation, and testing activities. It also acts as a reference document throughout the software development life cycle. Any future modifications or enhancements can be planned by referring to the defined requirements. Although Agile methodology allows evolving requirements, the SRS provides an initial baseline that is refined during sprint planning and reviews.

3.2 Scope

The scope of the Automated Banking Website includes providing online banking services to customers through a secure web-based platform. The system enables users to register, log in securely, view account details, check balance, transfer funds, pay bills, and access transaction history. Administrative functionalities such as user management, account monitoring, and report generation are also included. The system is intended for individual customers and bank administrators. Core banking back-end operations such as inter-bank settlements and ATM hardware integration are outside the scope of this project. The focus is on automating customer-facing banking services to enhance efficiency and accessibility.

3.3 Hardware Requirement / Software Requirement

Hardware Requirements

Component	Specification
Processor	Intel i3 or higher
RAM	Minimum 4 GB
Hard Disk	20 GB free space
Network	Stable Internet Connection

Software Requirements

Software	Description
Operating System	Windows / Linux
Web Browser	Google Chrome, Mozilla Firefox
Database	MySQL
Server	Apache / Tomcat

These requirements ensure smooth performance and secure handling of banking operations.

3.4 Tools

Various tools are used during the development of the Automated Banking Website to improve productivity and quality. Front-end technologies such as HTML, CSS, and JavaScript are used to create responsive user interfaces. Back-end development is carried out using server-side programming languages such as Java, PHP, or Python. MySQL is used as the database for storing user and transaction data.

Development and testing tools include Integrated Development Environments (IDE) such as Visual Studio Code or Eclipse, version control systems like Git, and testing tools for validating system functionality. Agile project management tools are used for sprint planning, progress tracking, and team collaboration.

3.5 Software Process Model

The project follows the Agile software process model. Agile emphasizes iterative development, continuous integration, and frequent stakeholder feedback. The system is divided into small modules, with each module developed during a sprint. At the end of every sprint, a working software increment is delivered and reviewed. This approach enables early defect detection, effective risk management, and improved customer satisfaction.

4. System Design

System Design is a critical phase of the software development life cycle in which the requirements identified during analysis are transformed into a detailed blueprint for building the Automated Banking Website.

This phase defines how the system will function internally and how different components will interact with each other. A well-planned system design ensures that the banking application is secure, scalable, reliable, and easy to maintain. In an Automated Banking Website, system design plays a vital role because the system handles sensitive financial data and real-time transactions.

Any design flaw may lead to data inconsistency, security risks, or system failure. Therefore, the design phase focuses on clearly defining system architecture, data structures, process flow, and user interaction before implementation begins.

The system design is broadly divided into logical design and physical design. Logical design describes what the system will do, while physical design explains how the system will be implemented using hardware and software resources.

4.1 Data Dictionary

A Data Dictionary is a centralized repository that contains detailed information about all data elements used in the system. It defines the meaning, purpose, and usage of data items such as user details, account information, and transaction records.

The data dictionary acts as a reference document for developers, testers, and database administrators. In the Automated Banking Website, the data dictionary ensures consistency in data usage across different modules. It helps avoid ambiguity, redundancy, and data conflicts.

By clearly defining each data element, the system maintains the accuracy and integrity of financial information. The data dictionary also supports easier database maintenance and future system enhancements.

4.2 Entity Relationship (ER) Diagram

The Entity Relationship (ER) Diagram represents the logical structure of the database used in the Automated Banking Website. It identifies key entities, their attributes, and the relationships among them. Core entities such as User, Account, Transaction, and Admin form the foundation of the banking system.

The ER diagram helps in organizing data efficiently and ensures proper relationships between different entities. For example, a user can have one or more bank accounts, and each account

can generate multiple transactions.

By clearly defining these relationships, the ER diagram supports database normalization and reduces data duplication. It is essential for designing a secure and well-structured database, which is crucial for maintaining accuracy and consistency in banking operations.

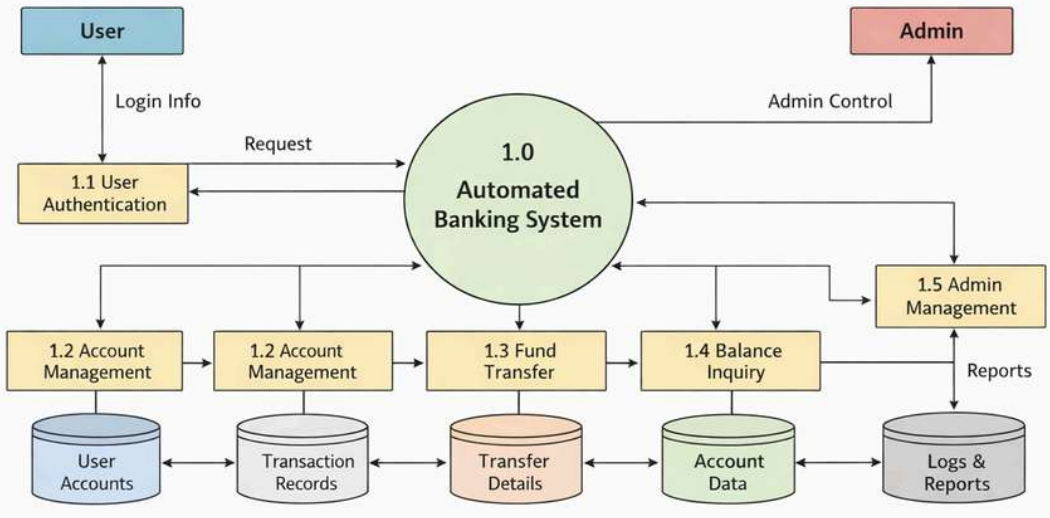
4.3 Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) illustrates how data flows through the system and how it is processed at different stages. It shows the movement of data from external entities, such as users, to internal processes and data stores.

In the Automated Banking Website, the DFD explains processes such as user login, transaction processing, balance inquiry, and data storage. The context-level DFD represents the entire banking system as a single process interacting with users and the database.

Lower-level DFDs decompose the system into smaller processes, providing a detailed understanding of system functionality.

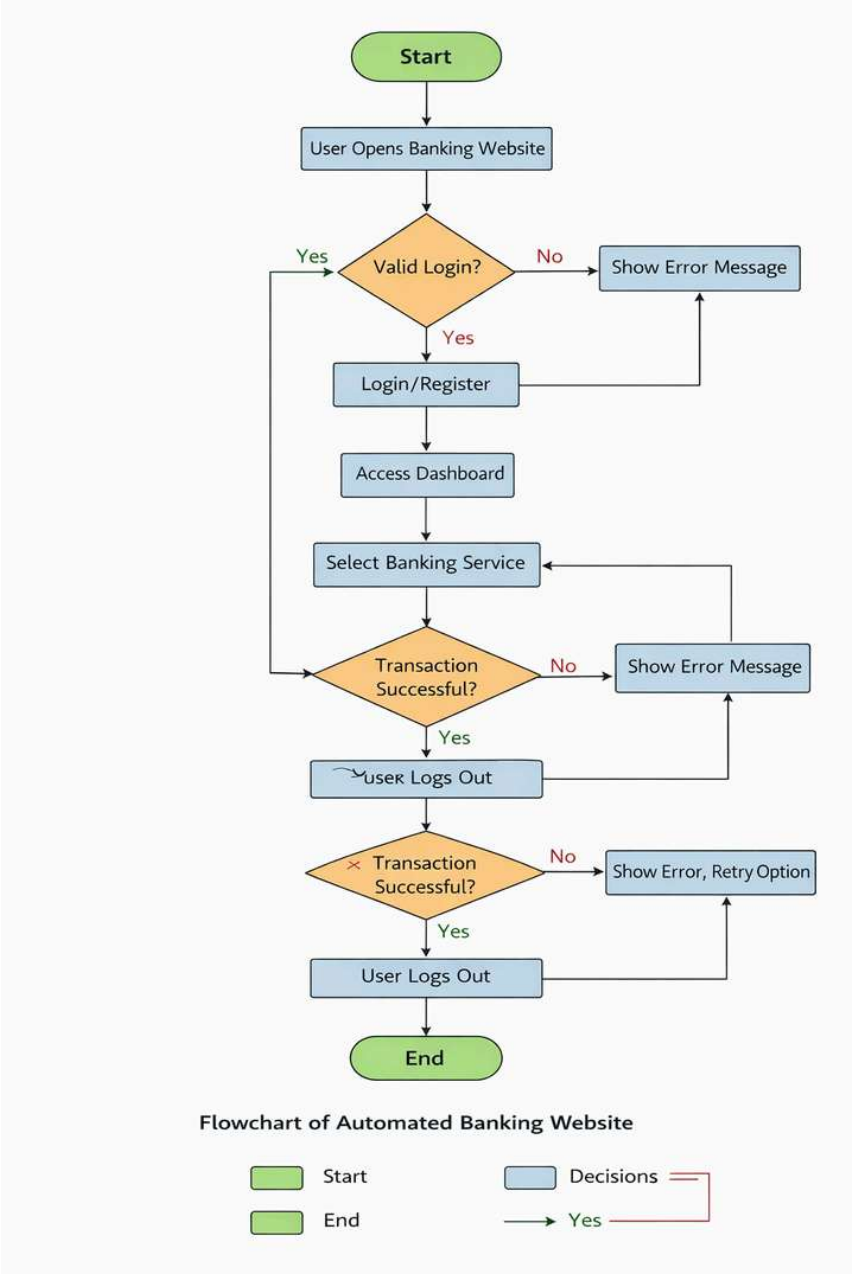
DFDs help visualize system operations, identify data dependencies, and ensure that all required processes are properly handled.



Data Flow Diagram of Automated Banking Website (Level 0 and Level 1)

4.4 System Flow Chart, Use Case Diagram, and Activity Diagram

The System Flow Chart represents the step-by-step execution of processes in the Automated Banking Website. It provides a clear view of control flow and logical sequence of operations within the system.

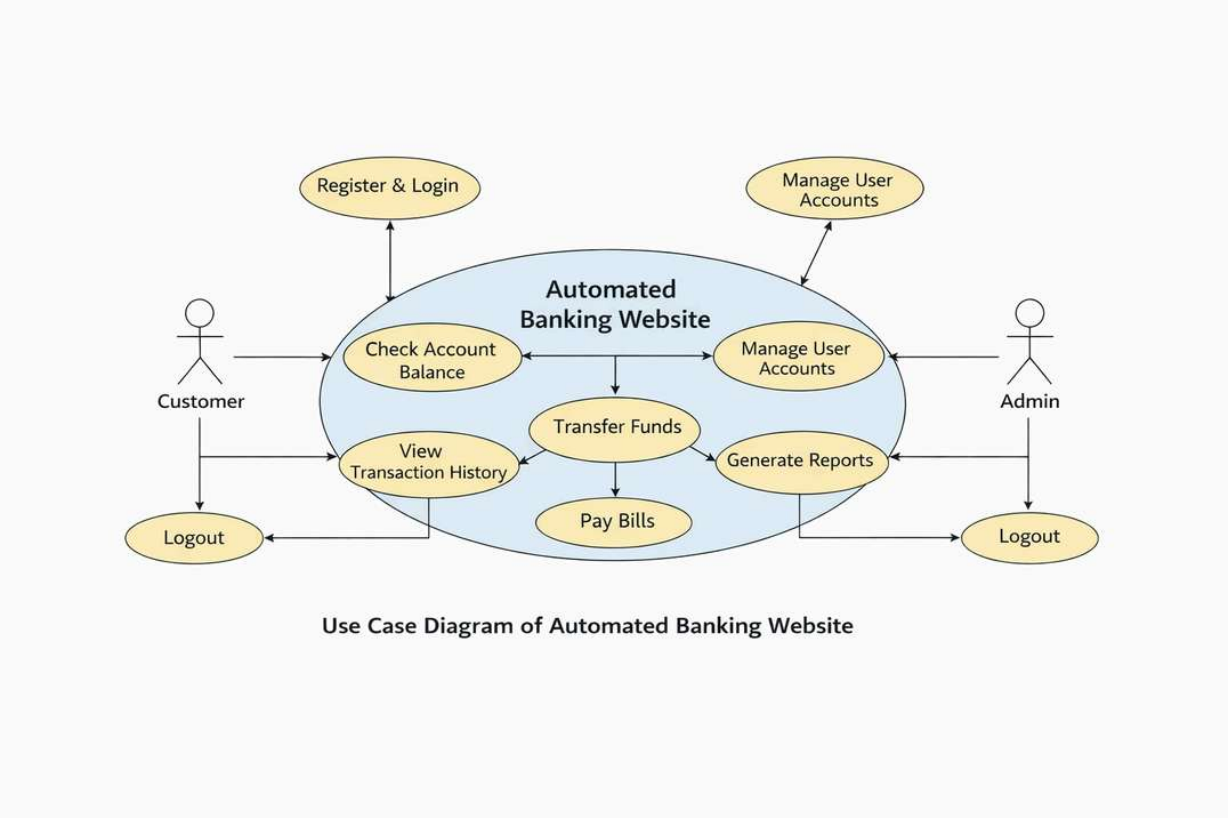


The Use Case Diagram focuses on user interaction with the system. It identifies different actors, such as customers and administrators, along with their possible actions. This diagram helps in

understanding functional requirements from the user's perspective.

The Activity Diagram describes the workflow of the system, including decision points and parallel activities. It provides a dynamic view of system behavior and helps in analyzing complex processes such as transaction validation and authorization.

Together, these diagrams provide a clear and complete understanding of system behavior, interaction, and workflow, making the design phase more effective and reducing errors during implementation.



Use Case Diagram of Automated Banking Website

5. Implementation

The implementation phase is a crucial stage of the software development life cycle in which the system design is converted into a fully functional Automated Banking Website. During this phase, all planned features and modules are developed using suitable programming technologies and integrated to function as a complete system. Implementation transforms theoretical models, diagrams, and design documents into actual working software.

As banking applications handle sensitive financial data, special emphasis is given to security, accuracy, reliability, and performance during implementation. Agile methodology is followed, allowing incremental development of the system. Each iteration focuses on developing specific functionalities, testing them, and refining them based on feedback. This approach helps in reducing risk and improving overall system quality.

The system is implemented using a modular architecture, where each module performs a specific function independently while interacting with other modules. This design makes the system easier to maintain, test, and enhance in the future.

Main goals of the implementation phase include:

- Developing a secure and reliable banking system
- Ensuring accurate processing of financial transactions
- Providing a smooth and user-friendly interface
- Efficient integration of frontend, backend, and database
- Supporting scalability and future enhancements

5.1 Program Code

Program code development is the core activity of the implementation phase. The code is written following standard coding practices to ensure clarity, security, and maintainability. The Automated Banking Website is divided into logical modules such as user authentication, account management, transaction processing, and administrative control.

While developing the program code, the following aspects are considered:

- Secure user authentication and authorization
- Input validation to prevent incorrect or malicious data
- Proper exception handling to avoid system crashes
- Transaction consistency to ensure correct debit and credit operations
- Secure database connectivity and data protection

The backend code manages business logic such as balance calculation, transaction verification, and data storage. The frontend code handles user interaction, including forms, navigation, and information display. Database operations are optimized to ensure fast response time and accurate data retrieval.

Agile development practices ensure frequent testing of code, enabling early detection and correction of errors. This results in a stable and high-quality banking application.

5.2 Output Screens

Output screens represent the visible interface of the Automated Banking Website and play a significant role in user experience. These screens are designed to be simple, clear, and easy to understand so that users with minimal technical knowledge can operate the system efficiently.

During screen design, the following factors are considered:

- Clean and consistent layout
- Clear labels and instructions
- Easy navigation between pages
- Quick system response
- Secure display of sensitive information

The system provides different output screens for various banking activities such as login, registration, dashboard, balance inquiry, fund transfer, and transaction history. An additional administrative interface is included for managing users and monitoring transactions.

Well-designed output screens enhance system usability, reduce user errors, and improve customer satisfaction. Thus, the implementation phase successfully delivers a secure, efficient, and user-friendly Automated Banking Website.

6. Testing

Testing is a critical phase in the software development life cycle that ensures the Automated Banking Website functions correctly, securely, and reliably. Since banking systems handle sensitive financial data and real-time transactions, thorough testing is essential to prevent errors that may lead to financial loss or security breaches. The primary objective of testing is to identify defects, verify system functionality, and ensure that the application meets all specified requirements.

In this project, testing is performed continuously by following Agile methodology. Each module is tested after development in every sprint, enabling early detection and correction of defects. This iterative testing approach enhances software quality and reduces overall development risk.

The testing phase covers both functional and non-functional aspects of the system. Functional testing verifies banking operations such as login, balance inquiry, fund transfer, and transaction history. Non-functional testing focuses on performance, security, usability, and reliability.

Major goals of the testing phase include:

- Verifying correct functionality of all system features
- Ensuring accuracy and consistency of financial transactions
- Detecting and fixing errors at an early stage
- Validating system security and access control
- Confirming compliance with user requirements

6.1 Test Data

Test data is prepared to simulate real-world banking scenarios. It includes both valid and invalid inputs to evaluate system behavior under different conditions. Proper test data helps in validating data input, error handling, and transaction accuracy.

Types of test data used include:

- Valid user credentials for successful login verification
- Invalid login details to check security and error handling
- Various transaction amounts to test fund transfer limits
- Incorrect inputs to validate system response

Well-planned test data ensures that the system handles all possible user actions safely and accurately.

Test Cases

Test Case ID	Test Scenario	Input Data	Expected Output
TC01	User Login	Valid username and password	Successful login
TC02	User Login	Invalid password	Error message displayed
TC03	New User Registration	Valid user details	Account created successfully
TC04	Balance Inquiry	Logged-in user request	Correct balance displayed
TC05	Fund Transfer	Valid amount within balance	Transaction successful
TC06	Fund Transfer	Amount greater than balance	Transaction failed with warning
TC07	Logout	User clicks logout	Session terminated successfully

6.2 Test Result

Test results are analyzed after executing all test cases to verify system performance and reliability. The Automated Banking Website is tested under normal and boundary conditions to ensure stable behavior. The system generates correct outputs for valid inputs and displays appropriate error messages for invalid actions.

The testing process confirms that:

- All banking functions operate as intended
- Data integrity is maintained during transactions
- Security mechanisms prevent unauthorized access
- System performance remains stable under expected load

After successful testing, the system is considered reliable, secure, and ready for deployment. Continuous testing in Agile development ensures that the system remains robust even when new

features or enhancements are introduced.

Test Results Summary

Test Case ID	Actual Result	Status
TC01	Login successful	Pass
TC02	Error message shown	Pass
TC03	Account created	Pass
TC04	Correct balance displayed	Pass
TC05	Amount transferred successfully	Pass
TC06	Error message displayed	Pass
TC07	User logged out safely	Pass

7. User Manual

The User Manual provides clear and simple guidelines for end users to operate the Automated Banking Website effectively. It is intended for users with basic computer knowledge and does not require any technical background. The main purpose of the user manual is to help users understand how to access banking services, perform transactions, and use system features safely and efficiently.

A well-prepared user manual increases user confidence, reduces errors, and minimizes the need for external support. In an online banking system, proper guidance is essential to ensure secure and correct usage of financial services. The manual explains system functionality in a step-by-step manner, enabling users to navigate the application easily and complete tasks without confusion.

7.1 How to Use Project Guidelines

To use the Automated Banking Website, users should follow a simple and secure process. First, the user opens the banking website using a supported web browser. New users must complete the registration process by entering valid personal and account details. Existing users can directly log in using their registered username and password. After successful login, the user is redirected to the dashboard, where all available banking services are displayed. From the dashboard, users can select services such as checking account balance, transferring funds, paying bills, or viewing transaction history. Each operation provides clear instructions to guide users throughout the process. After completing all required transactions, users are advised to log out properly to ensure account security. Following these guidelines ensures safe and efficient usage of the banking system.

7.2 Screen Layouts and Description

The screen layout of the Automated Banking Website is designed to be user-friendly, clean, and consistent. Each screen contains clearly labeled fields, buttons, and navigation options, allowing users to understand system functionality easily. Important information is displayed prominently, while sensitive data is protected and shown only when required.

The login screen enables secure entry of user credentials. The dashboard screen provides an overview of account details and available services. Transaction-related screens guide users step by step through financial operations. Error messages and confirmation notifications are displayed clearly to inform users about the status of their actions. The simple and intuitive screen design minimizes user errors, improves usability, and enhances the overall banking experience, making the system easy to operate for users of all age groups.

8. Project Applications and Limitations

The Automated Banking Website is designed to provide efficient, secure, and user-friendly online banking services. The system can be applied in various real-world banking scenarios; however, like any software system, it also has certain limitations. Understanding both applications and limitations helps in evaluating the effectiveness of the project and planning future improvements.

8.1 Project Applications

The Automated Banking Website has wide applicability in the modern banking environment. It supports digital transformation by automating routine banking operations and reducing dependency on physical branches.

Major applications of the project include:

- Providing 24×7 online banking services to customers
- Enabling secure balance inquiry and account management
- Supporting online fund transfer and digital payments
- Reducing manual workload and operational cost for banks
- Improving transaction speed and accuracy
- Enhancing customer convenience and satisfaction
- Supporting centralized data management and monitoring
- Allowing administrators to manage users and transactions efficiently

The system is suitable for small, medium, and large banking institutions that aim to improve service delivery through digital platforms.

8.2 Project Limitations

Despite its advantages, the Automated Banking Website has certain limitations that need to be considered. These limitations are common in most web-based banking systems and can be addressed through future enhancements.

Key limitations of the project include:

- Dependence on a stable internet connection
- Risk of cybersecurity threats such as hacking or phishing
- Limited functionality during server downtime
- Performance issues under extremely high user load
- Requirement of regular system maintenance and updates
- Dependence on user awareness for safe password practices

9. Conclusion and Future Enhancement

Conclusion

The Automated Banking Website project successfully demonstrates the use of modern web technologies and Agile methodology to develop a secure, reliable, and user-friendly online banking system. The project overcomes the limitations of traditional banking systems by automating essential banking operations such as user authentication, account management, balance inquiry, fund transfer, and transaction monitoring.

By adopting Agile development practices, the project ensures flexibility, continuous improvement, and early defect detection. Iterative development and regular testing result in a stable system that effectively meets user requirements. The modular system architecture enhances maintainability and supports future expansion.

The system improves customer convenience by providing 24×7 access to banking services and reduces manual workload for banking staff. Secure authentication mechanisms and controlled access protect sensitive financial data. Overall, the project achieves its objective of delivering an efficient digital banking solution suitable for modern banking environments.

Future Enhancement

Although the Automated Banking Website fulfills current banking requirements, there is scope for further improvement and expansion. Future enhancements can make the system more advanced, secure, and user-centric.

Possible future enhancements include:

- Development of mobile applications for Android and iOS platforms
- Integration of biometric authentication such as fingerprint or facial recognition
- Implementation of AI-based fraud detection and risk analysis
- Support for multiple languages to improve accessibility
- Integration with advanced payment gateways and digital wallets
- Performance optimization for high traffic conditions
- Enhanced reporting and analytics features for administrators

These enhancements will further strengthen the system, improve security, and enhance user experience, making the Automated Banking Website more adaptable to future technological advancements.

10. Bibliography & References

1. Pressman, R. S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill Education.
2. Sommerville, I., *Software Engineering*, Pearson Education.
3. Schwaber, K. and Sutherland, J., *The Scrum Guide*, Scrum.org.
4. IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*.
5. IEEE Std 1016-2009, *IEEE Standard for Information Technology – Systems Design*.
6. Kaur, K., *Agile Software Development*, Oxford University Press.
7. W3C, *HTML, CSS and Web Standards Documentation*.
8. Oracle, *Database Concepts and SQL Documentation*.
9. OWASP Foundation, *Web Application Security Guidelines*.
10. TutorialsPoint, *Online Banking System and Web Development Concepts*.