Automate Banking Website Using Agile Methodology

(Java – Console Based Application)

---

1. Abstract

The banking sector plays a crucial role in the financial stability and economic growth of a country. With the increase in the number of customers and transactions, traditional manual banking systems have become inefficient, time-consuming, and prone to errors. Customers today expect quick access to banking services, accurate transaction processing, and secure management of account details. To overcome these challenges, automated banking systems are widely adopted.

This project focuses on the design and development of an **Automate Banking Website using Agile methodology**, implemented as a **Java-based console application**. The proposed system allows users to create bank accounts, deposit and withdraw money, transfer funds, and view account details through a menu-driven console interface. An admin role is responsible for managing customer accounts and monitoring transactions. The console-based approach makes the system lightweight, easy to understand, and suitable for academic learning and small-scale banking simulations.

Agile methodology is used in this project to ensure flexibility, faster development, and continuous improvement. The development process is divided into short iterations called sprints, where each sprint delivers a functional banking module such as account management, transaction handling, or balance enquiry. Continuous testing and feedback during each sprint help identify errors early and improve system reliability.

The project is implemented using the Java programming language due to its object-oriented features, platform independence, security, and robustness. Various test cases are executed to validate system functionality and accuracy. This project demonstrates the practical application of Agile methodology in developing a banking automation system and provides scope for future enhancements such as database integration, graphical user interface, and online banking facilities.

---

2. Introduction

2.1 Introduction

An Automated Banking System is a software application that enables banks to manage customer accounts and transactions efficiently. It simplifies banking operations by providing a centralized system for handling deposits, withdrawals, fund transfers, and account records. Automation reduces manual effort and improves accuracy and service quality.

2.2 Problem Identification

Manual or semi-automated banking systems often lead to data inconsistency, calculation errors, difficulty in tracking transaction history, and security risks. As the number of customers increases, maintaining records manually becomes complex and inefficient.

2.3 Need of the Project

There is a strong need for an automated banking system that can manage customer accounts and transactions efficiently, reduce manual work, and improve accuracy. A Java-based console application provides a simple, cost-effective solution for banking automation, especially for educational and training purposes.

2.4 Project Scheduling

The project is developed using **Agile methodology** and divided into the following phases:

- Requirement Analysis and Planning
- System Design
- Implementation
- Testing
- Documentation

Each phase is completed in short iterations to allow flexibility, continuous feedback, and improvement.

2.5 Objectives

- To develop an automated banking system using Java
- To apply Agile methodology in software development

● To manage bank accounts and transactions efficiently
● To understand object-oriented programming concepts

---

3. Software Requirement Specification (SRS)

3.1 Purpose

The purpose of this Software Requirement Specification (SRS) is to describe the functional and non-functional requirements of the Automated Banking System. It serves as a reference document for understanding system behavior, constraints, and overall functionality.

3.2 Scope

The scope of the project includes basic banking operations such as account creation, deposit, withdrawal, fund transfer, and viewing account details. Advanced features like online banking, ATM integration, and real-time transaction processing are outside the current scope.

3.3 Hardware / Software Requirements

Hardware Requirements:

● Processor: Intel i3 or above
● RAM: Minimum 4 GB
● Hard Disk: Minimum 10 GB free space

Software Requirements:

● Operating System: Windows 10 / Windows 11
● Programming Language: Java (JDK 8 or above)
● IDE: Eclipse IDE

3.4 Tools

- Java Development Kit (JDK)
- Eclipse IDE
- GitHub (for report hosting)

3.5 Software Process Model

The **Agile Software Development Model** is used in this project. Agile emphasizes iterative development, customer collaboration, continuous testing, and quick response to change.

---

4. System Design

4.1 Data Dictionary

Field Name	Description
AccountNumber	Unique bank account number
CustomerName	Name of the account holder
AccountType	Savings / Current
Balance	Available account balance
TransactionID	Unique transaction identifier
Amount	Transaction amount
TransactionType	Deposit / Withdrawal / Transfer

4.2 ER Diagram

The Entity Relationship (ER) diagram consists of entities such as **Customer**, **Account**, **Transaction**, and **Admin**. Each account is associated with a customer, and each transaction is linked to an account.

4.3 Data Flow Diagram (DFD)

The Data Flow Diagram illustrates the flow of data between the user and the Automated Banking System. It includes processes such as account creation, money deposit, withdrawal, fund transfer, and balance enquiry.

4.4 Use Case Diagram

Actors involved in the system are **User** and **Admin**. The main use cases include Create Account, Deposit Money, Withdraw Money, Transfer Funds, View Account Details, and Manage Accounts.

---

5. Implementation

5.1 Program Code

The Automated Banking System is implemented using the Java programming language and follows a menu-driven console approach. Object-oriented concepts such as classes, objects, encapsulation, inheritance, and collections are used to manage banking data efficiently.

5.2 Output Screens

The application displays text-based menus on the console, allowing users to perform banking operations and view transaction confirmations or error messages.

---

6. Testing

6.1 Test Data

Sample account data and transaction values are used to test functionalities such as account creation, deposit, withdrawal, and fund transfer. Invalid inputs are also tested to ensure proper validation and error handling.

6.2 Test Result

All test cases were executed successfully. The system produced correct results for valid transactions and displayed appropriate error messages for invalid operations such as insufficient balance or invalid account numbers.

---

7. User Manual

7.1 How to Use Project Guidelines

1. Run the Java application
2. Select the required option from the main menu
3. Enter account and transaction details as prompted
4. View transaction confirmation on the console
5. Exit the application safely

7.2 Screen Layouts and Description

The application uses a simple console-based interface with clearly labeled menu options for ease of use and smooth navigation.

---

8. Project Applications and Limitations

Applications

- Small banking institutions
- Educational institutions
- Training and learning purposes

Limitations

- Console-based user interface
- No online banking features

● No database connectivity
● Limited security features

---

9. Conclusion and Future Enhancement

The Automated Banking System developed using Java and Agile methodology successfully demonstrates how banking operations can be automated efficiently. The project enhanced understanding of Agile practices, system design, and Java programming.

Future enhancements may include database integration, graphical user interface development, online banking services, enhanced security mechanisms, and mobile application support.

---

10. Bibliography & References

● Agile Manifesto
● Java Programming Documentation
● Software Engineering Textbooks