

Software Requirements Specification

For

**Automated Banking System**

Version 1.0 Approved

**Prepared by: Devansh Dixit**

**HCL Training**

Date: 03/01/2026

Submitted in partial fulfillment

Of the requirements of

Software Engineering(Agile Methodology)

## **Table of Contents**

1. **Abstract**
2. **Introduction**
  - 2.1 Introduction
  - 2.2 Problem Identification
  - 2.3 Need of the Project
  - 2.4 Project Scheduling
  - 2.5 Objectives
3. **Software Requirement Specification (SRS)**
  - 3.1 Purpose
  - 3.2 Scope
  - 3.3 Hardware Requirement / Software Requirement
  - 3.4 Tools
  - 3.5 Software Process Model
4. **System Design**
  - 4.1 Data Dictionary
  - 4.2 ER Diagram
  - 4.3 Data Flow Diagram (DFD)
  - 4.4 System Flow Chart / Object Diagram / Class Diagram / State Diagram / Activity Diagram / Use Case Diagram
5. **Implementation**
  - 5.1 Program Code
  - 5.2 Output Screens
6. **Testing**
  - 6.1 Test Data
  - 6.2 Test Result
7. **User Manual**
  - 7.1 How to Use Project Guidelines
  - 7.2 Screen Layouts and Description
8. **Project Applications and Limitations**
9. **Conclusion and Future Enhancement**
10. **Bibliography and References**

## Abstract

The Automated Banking System (ABS) is a software application designed to manage banking operations efficiently, securely, and accurately. Traditional banking relies heavily on manual processing, leading to delays, errors, and inefficiencies in handling transactions, account management, and customer service. With the rapid growth of digital banking, there is a need for an automated solution that ensures fast and accurate banking services.

This system allows customers to perform banking operations such as account creation, balance enquiry, deposit, withdrawal, fund transfer, and mini-statement generation. Administrators can manage customer accounts, monitor transactions, generate reports, and maintain security protocols. The system is designed to reduce human errors, improve transaction speed, and provide 24/7 accessibility to banking services.

The Automated Banking System is developed using the **Agile methodology**, which divides development into iterative sprints. This ensures early delivery of functional modules, continuous testing, and incorporation of user feedback at every stage. Security and data integrity are prioritized, ensuring confidential information such as account details, passwords, and transactions are protected.

By implementing this system, banks can enhance operational efficiency, reduce paperwork, and provide better customer satisfaction. Future enhancements may include mobile banking integration, biometric authentication, and AI-based fraud detection. This project demonstrates how Agile practices can help develop a secure, reliable, and user-friendly banking solution.

## 2.Introduction

The **Automated Banking System (ABS)** is a software application designed to digitize banking operations, including account management, transactions, and reporting. Traditional banking relies on manual processes, which are time-consuming, error-prone, and inefficient. With the rise of digital banking, there is a need for a system that automates core banking tasks while ensuring security, accuracy, and real-time access for customers and administrators.

The ABS enables customers to manage their accounts, perform transactions, and access banking services online. Administrators can manage accounts, monitor transactions, and generate reports efficiently. By using Agile methodology, the system is developed in iterative sprints, ensuring early delivery of modules, continuous testing, and incorporation of user feedback.

### 2.1 Introduction

Banking operations involve handling large volumes of customer data, managing financial transactions, and maintaining accurate account information. Manual management of these processes is inefficient and increases the risk of errors.

The Automated Banking System provides a **centralized, digital platform** where:

- Customers can register, log in, view account details, deposit and withdraw funds, transfer money, and generate mini-statements.
- Administrators can manage customers, monitor transactions, and generate reports.

This system reduces dependency on physical branch visits and allows **24/7 access** to banking operations.

### 2.2 Problem Identification

Key problems in traditional banking include:

- Long queues for simple transactions such as deposits or withdrawals.
- Manual record-keeping, which may cause errors in balances or transaction history.
- Difficulty in generating accurate financial reports quickly.
- Time-consuming fund transfers between accounts.
- Limited access to banking services outside business hours.

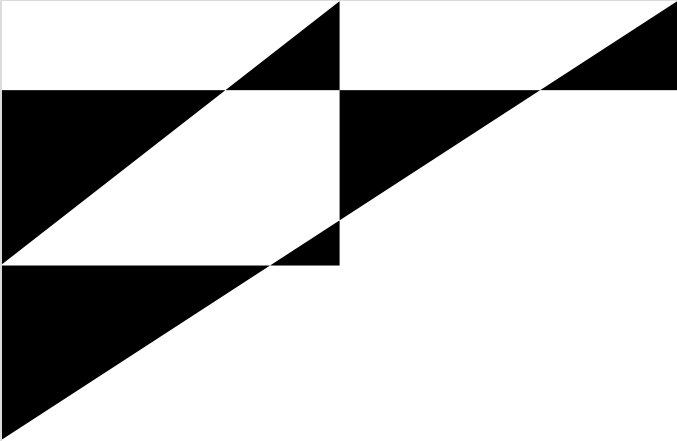
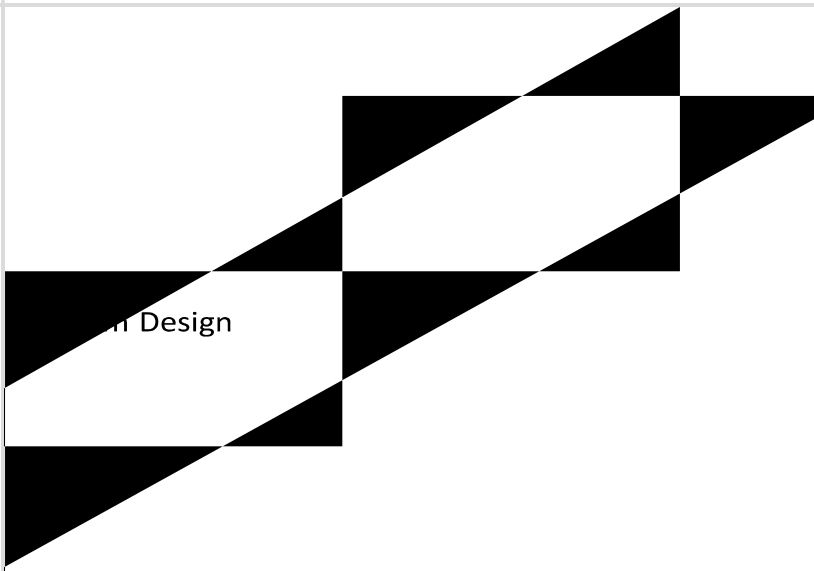
### 2.3 Need of the Project

The Automated Banking System is needed to:

- **Automate banking operations** such as account management, transactions, and reporting.
- **Reduce errors** in financial transactions and account balances.
- **Provide customers with 24/7 access** to their accounts and services.
- **Improve efficiency** in administrative and operational tasks.
- **Enhance security** by providing user authentication and controlled access.

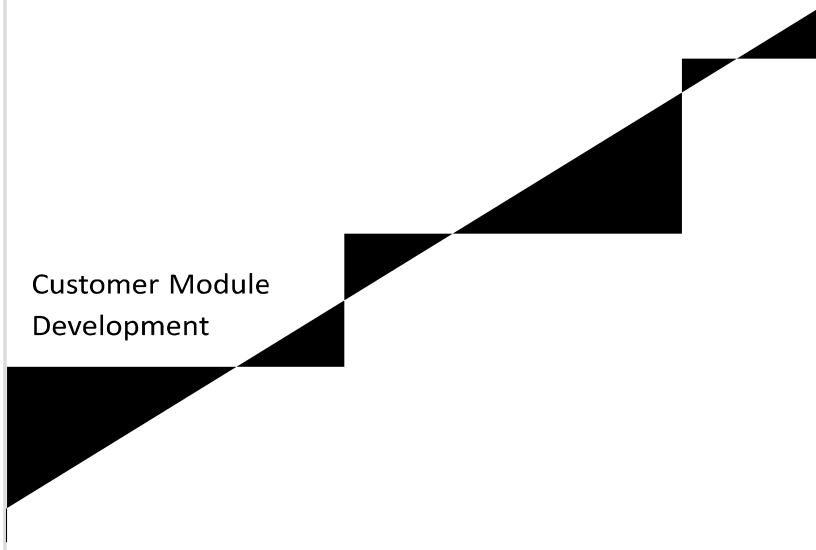
## 2.4 Project Scheduling

The project follows the **Agile methodology** and is divided into multiple **sprints** to ensure iterative development:

| Sprint No.  | Sprint Name   |
|---|---|
|  | Requirement Analysis  |
| 2   |  System Design |

3

Customer Module  
Development



4

Transaction Module

|   |                      |
|---|----------------------|
| 5 | Admin Module         |
| 6 | Testing & Deployment |

## 2.5 Objectives

- To develop a secure and efficient Automated Banking System.
- To allow customers to manage accounts and transactions online.
- To reduce manual errors and improve operational efficiency.
- To enable administrators to monitor accounts and generate reports.
- To provide real-time balance updates and transaction confirmations.
- To implement the system using **Agile methodology** for iterative and flexible development.

## 3. Software Requirement Specification (SRS)

The Software Requirement Specification defines the functional and non-functional requirements of the Automated Banking System. It serves as a reference for developers, testers, and stakeholders to ensure the system meets the banking operations requirements efficiently and securely.

### 3.1 Purpose

The purpose of this SRS is to clearly define the requirements of the Automated Banking System (ABS) to ensure:

- Customers can perform banking operations such as deposits, withdrawals, fund transfers, balance enquiry, and mini-statement generation.
- Administrators can manage accounts, monitor transactions, and generate reports efficiently.
- The system ensures data security, accuracy, and integrity.
- Agile methodology is followed to allow iterative development and continuous improvement based on user feedback.

The document also helps in testing, deployment, and future maintenance of the system.

### 3.2 Scope

The scope of the Automated Banking System includes:

- **Customer Module:**
  - Registration and login
  - View account details
  - Deposit and withdraw funds
  - Fund transfers between accounts
  - Mini-statement generation
- **Admin Module:**
  - Manage customer accounts (add, update, delete)
  - Monitor transactions
  - Generate daily, monthly, and annual reports
  - Ensure system security and user authentication
- **Security:**
  - Password protection for accounts
  - Restricted access for administrators
  - Data validation for transactions
- **System Benefits:**
  - Reduces manual workload
  - Provides 24/7 online access
  - Ensures accurate and up-to-date banking data
  -

### 3.3 Hardware Requirement / Software Requirement (Minimum)

Hardware Requirements:

| Component | Specifica   |
|-----------|-------------|
| Processor | Intel i3 or |
| RAM       | 4 GB or h   |



|           |            |
|-----------|------------|
| Hard Disk | 20 GB free |
| Monitor   | 15-inch    |
| Internet  | Required   |

Software Requirements:

| Software         | Specifica |
|------------------|-----------|
| Operating System | Windows   |
| Database         | MySQL     |
| Backend Language | Java / No |
| Frontend         | HTML, CS  |

|         |           |
|---------|-----------|
| IDE     | Eclipse , |
| Browser | Chrome ,  |

### 3.4 Tools

The following tools are used for the development of the Automated Banking System:

- **Frontend Development:** HTML, CSS, JavaScript
- **Backend Development:** Java / Node.js
- **Database Management:** MySQL
- **IDE:** Eclipse or Visual Studio Code
- **Version Control:** Git (for source code management) **Server (if**
- **web-based):** Apache / Tomcat


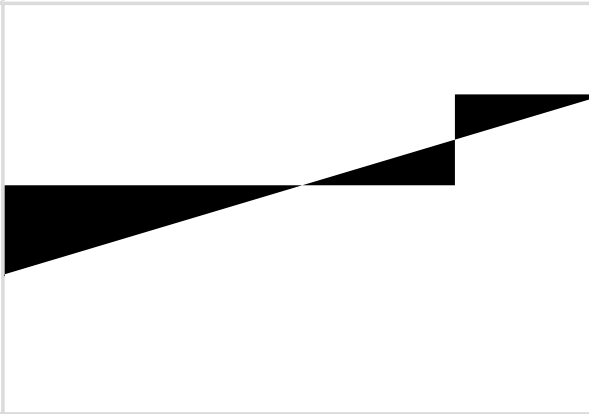
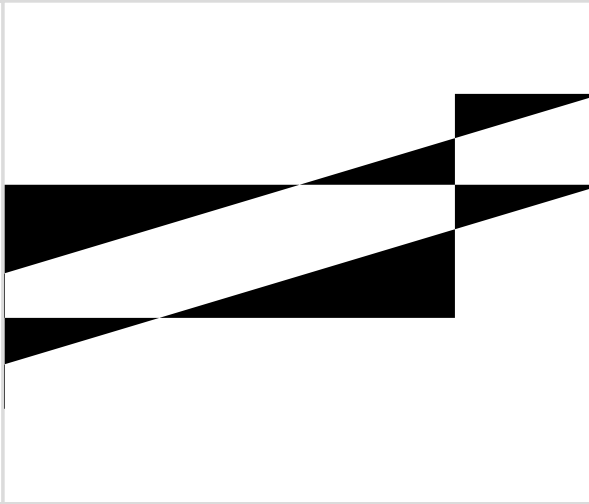
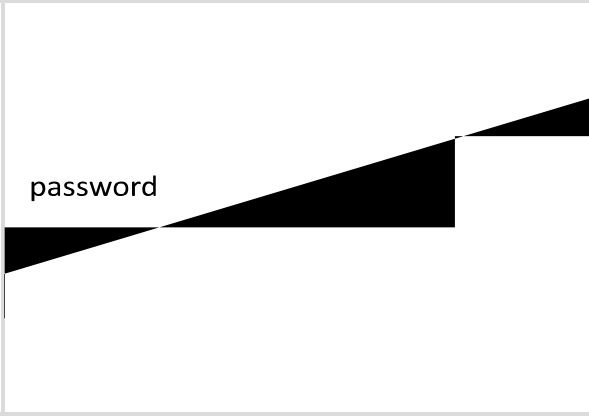
These tools ensure the system is **secure, scalable, maintainable, and user-friendly**.

## 4. System Design

System design defines the structure, components, and workflow of the Automated Banking System. Proper design ensures the system is efficient, scalable, and secure while providing all banking functionalities.

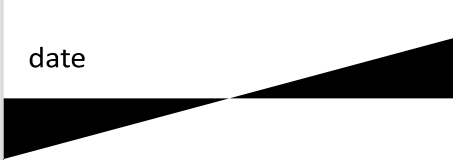

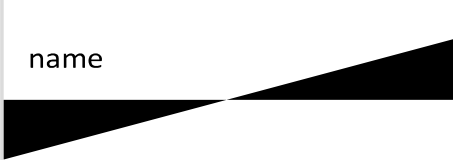
### 4.1 Data Dictionary

The Data Dictionary lists all major entities and their attributes used in the system:

| Entity  | Attribute  |
|---|--|
| <br>customer | customer_id  |
|   |             |
|   |            |
|   | password  |

|         |              |
|---------|--------------|
|         | account_no   |
|         | balance      |
| Account | account_no   |
|         | customer_id  |
|         | account_type |

|             |                |
|-------------|----------------|
| Transaction | transaction_id |
|             |                |
|             | type           |
|             | amount         |

|       |   |
|-------|---|
|       | <br>date     |
| Admin | <br>admin_id |
|       | <br>name    |
|       | email   |
|       | password  |

## 4.2 ER Diagram (Entity-Relationship Diagram)

The **ER Diagram** shows relationships between the main entities in the system:

one-to-one (1:1)



one-to-many (1:N)



many-to-one (N:1)

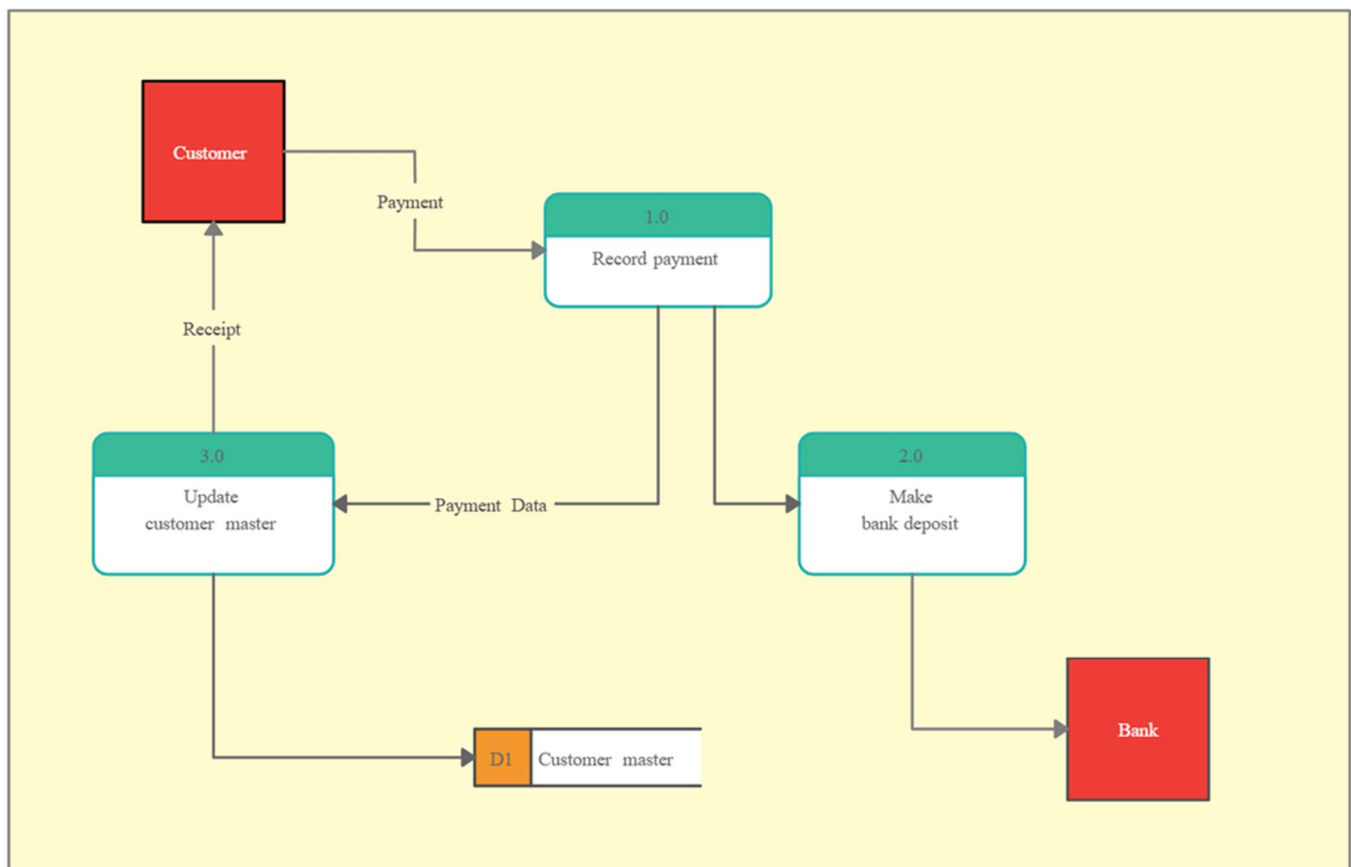


many-to-many (M:N)



### 4.3 Data Flow Diagram (DFD)

The **DFD** represents how data flows between users and the Automated Banking System:



## 5. Implementation

Implementation is the phase where the **system design is transformed into a working software application**. In the Automated Banking System, each module—such as customer management, transaction processing, account management, and admin operations—is developed using **Java** (backend) and **HTML/CSS/JavaScript** (frontend). The system is modular, user-friendly, and secure.

### 5.1 Program Code (Sample)

**Customer Class (Java)** public

```

class Customer {
    private int
    customerId;
    private String
    name;
    private String email;
    private String password;
    private double balance;
  
```

```

    public Customer(int customerId, String name, String email, String password, double balance) {
        this.customerId = customerId;
  
```



```
        this.name = name;

this.email = email;

this.password = password;

this.balance = balance;

    }
```

```
public void displayCustomerDetails() {

    System.out.println("Customer ID: " + customerId);

    System.out.println("Name: " + name);

    System.out.println("Email: " + email);

    System.out.println("Balance: " + balance);

}
```

```
public void deposit(double amount) {

balance += amount;

    System.out.println("Deposited: " + amount + " | New Balance: " + balance);

}
```

```
public void withdraw(double amount) {

if(amount <= balance) {        balance -=

amount;

    System.out.println("Withdrawn: " + amount + " | New Balance: " + balance);

} else {

    System.out.println("Insufficient balance!");

}

}

}
```

**Transaction Class (Java)** public

```
class Transaction {    private
int transactionId;    private int
accountNo;    private String
type;    private double
amount;    private String date;

    public Transaction(int transactionId, int accountNo, String type, double amount, String date) {
this.transactionId = transactionId;        this.accountNo = accountNo;        this.type = type;
this.amount = amount;        this.date = date;
    }

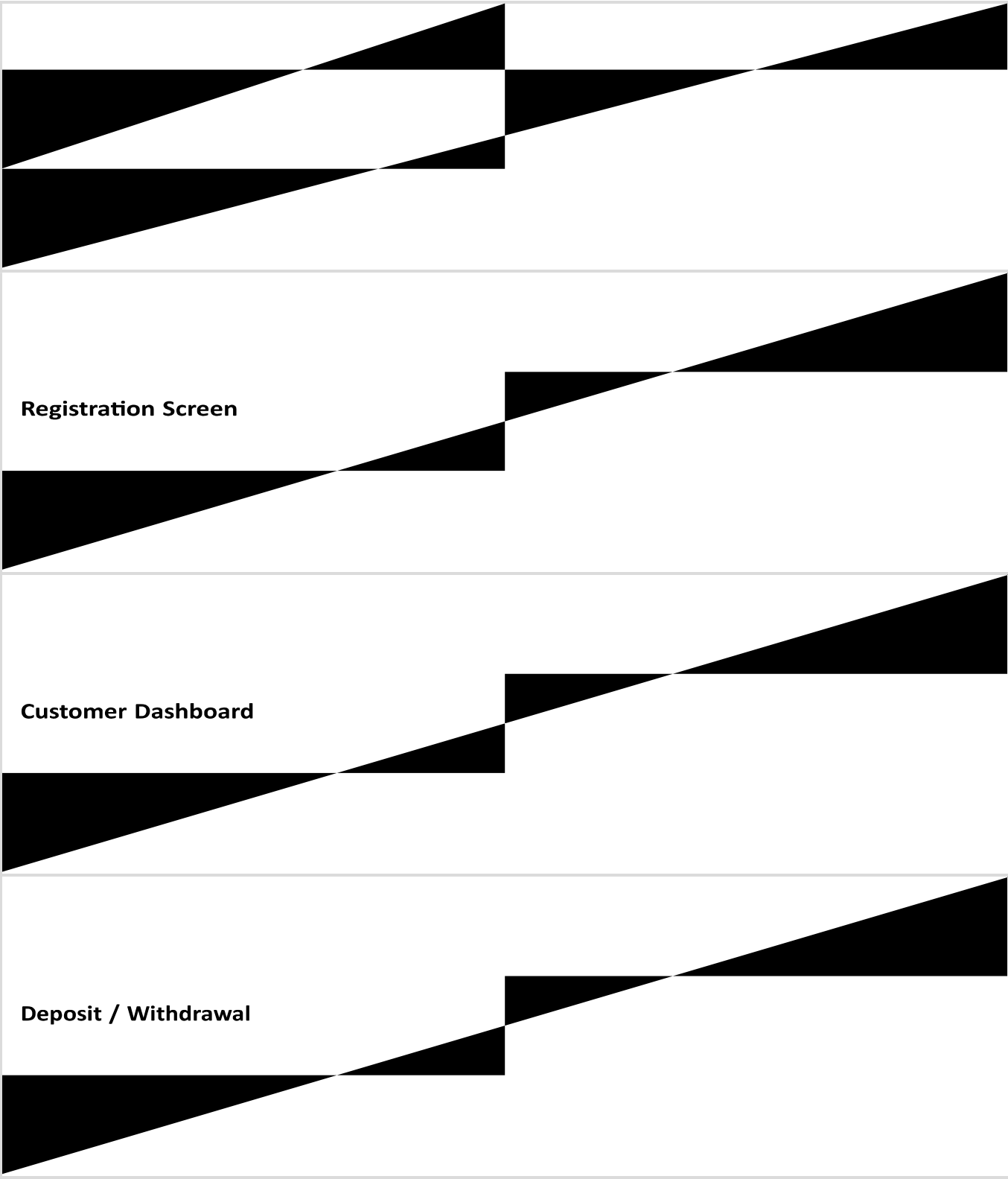
    public void displayTransaction() {
        System.out.println("Transaction ID: " + transactionId);
        System.out.println("Account No: " + accountNo);
        System.out.println("Type: " + type);
        System.out.println("Amount: " + amount);
        System.out.println("Date: " + date);
    }
}
```

**Note:** Full implementation includes **database connectivity** for account info and transaction records, as well as a **web interface** for user interaction.

**5.2 Output Screens**

The Automated Banking System has several key screens:

| Screen | Descripti |
|--------|-----------|
|--------|-----------|



Input em  
Provides

New user

Displays  
operation

Input am  
confirma

|   |                        |
|---|------------------------|
| <b>Fund Transfer</b>                        | Transfer m<br>and upda |
| <b>Transaction History / Mini-Statement</b> | Displays<br>account    |
| <b>Admin Dashboard</b>                      | Allows ac<br>and gene  |

6. Testing

Testing ensures that the **Automated Banking System (ABS)** works correctly, meets the functional requirements, and provides secure, reliable banking services. Both **functional testing** (checking modules like login, registration, and transactions) and **non-functional testing** (security, performance, usability) are performed.

The testing phase helps identify and correct errors, verify system accuracy, and ensure smooth operation.

6.1 Test Data

The following test data is used to validate different modules of the system:

| Test Case ID | Module                   |
|--------------|--------------------------|
|              | Customer<br>Registration |

TC02

Customer  
Registration

TC03

Login

TC04

Login

TC05

Deposit

TC06

Withdrawal

TC07

Withdrawal

TC08

Fund Transfer

TC09

Fund Transfer



TC10

Transaction History



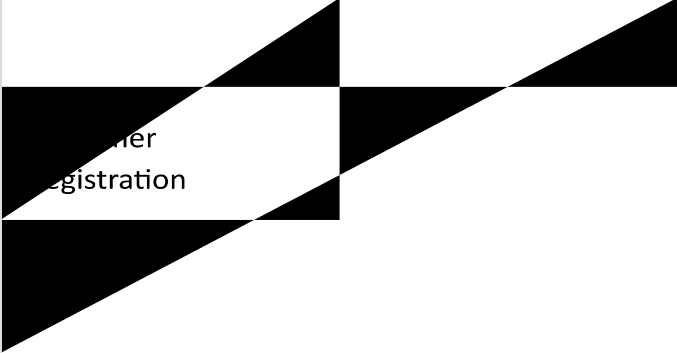
TC11

Admin Login

|      |              |
|------|--------------|
| TC12 | Admin Module |
| TC13 | Admin Module |

## 6.2 Test Results

After executing all test cases, the observed results are summarized below:

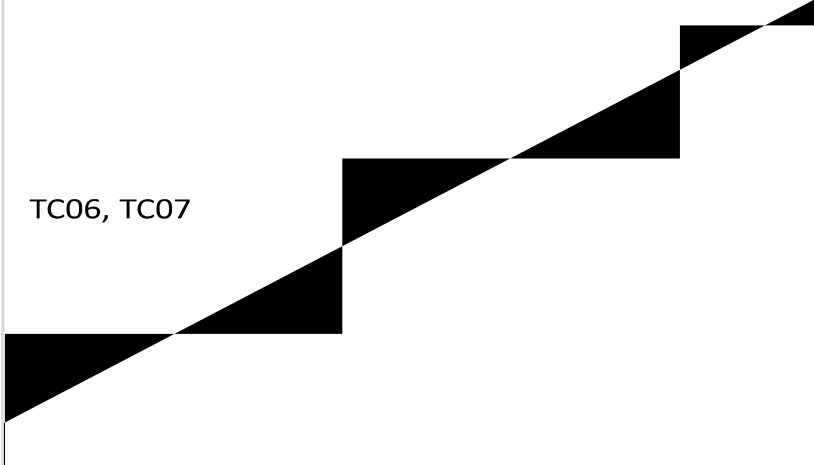
| Module  | Test Case ID |
|---|--------------|
|  | TC01, TC02   |

Login

, TC04

Deposit

TC05

|                            |  |
|----------------------------|--|
| <p>Withdrawal</p>          |  <p>TC06, TC07</p> |
| <p>Fund Transfer</p>       | <p>TC08, TC09</p>  |
| <p>Transaction History</p> | <p>TC10</p>  |

|              |                  |
|--------------|------------------|
| Admin Module | TC11, TC12, TC13 |
|--------------|------------------|

## 7. User Manual

The **User Manual** provides detailed instructions for using the Automated Banking System (ABS). It helps **customers, admins**, and bank staff to navigate the system effectively, perform banking operations, and access features without errors.

### 7.1 How to Use Project Guidelines

#### Step-by-Step Instructions for Users:

##### 1. Launching the System

- Open the Automated Banking System in a web browser or application interface. Ensure you
- have internet access if the system is online.

##### 2. User Registration (New Customers)

- Click on “Register” or “Sign Up.”
- Enter your **name, email, password, and contact information**.
- Click **Submit** to create your account.
- You will receive a confirmation message after successful registration.

##### 3. Login

- Registered users enter their **email** and **password** on the login page.
- Admins have separate login credentials. Click
- **Login** to access the dashboard.

##### 4. Customer Dashboard

- **View Account Details:** Check account number, balance, and account type.
- **Deposit Money:** Enter the amount to deposit. Confirm transaction.
- **Withdraw Money:** Enter the amount to withdraw. Confirm transaction.
- **Fund Transfer:** Enter sender and receiver account numbers, amount, and confirm. **Transaction History**
- **/ Mini-Statement:** View last few transactions.

##### 5. Admin Dashboard

- Manage customer accounts: Add, update, delete accounts.
- Monitor all transactions.
- Generate reports: Daily, monthly, yearly summaries. Ensure
- system security and verify operations.

##### 6. Logout

- Always log out after completing operations to maintain account security.

### 7.2 Screen Layouts and Description

| Screen | Description |
|--------|-------------|
|--------|-------------|

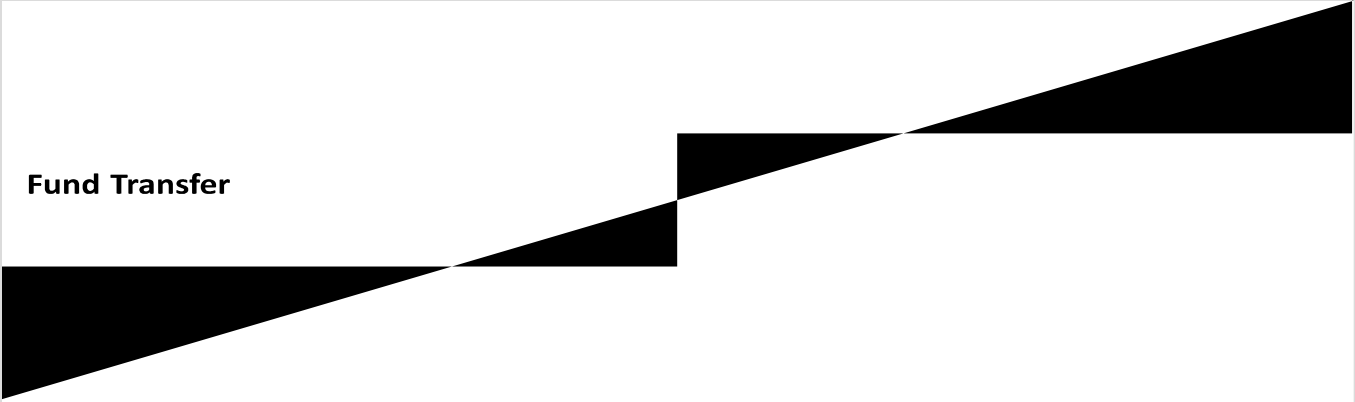
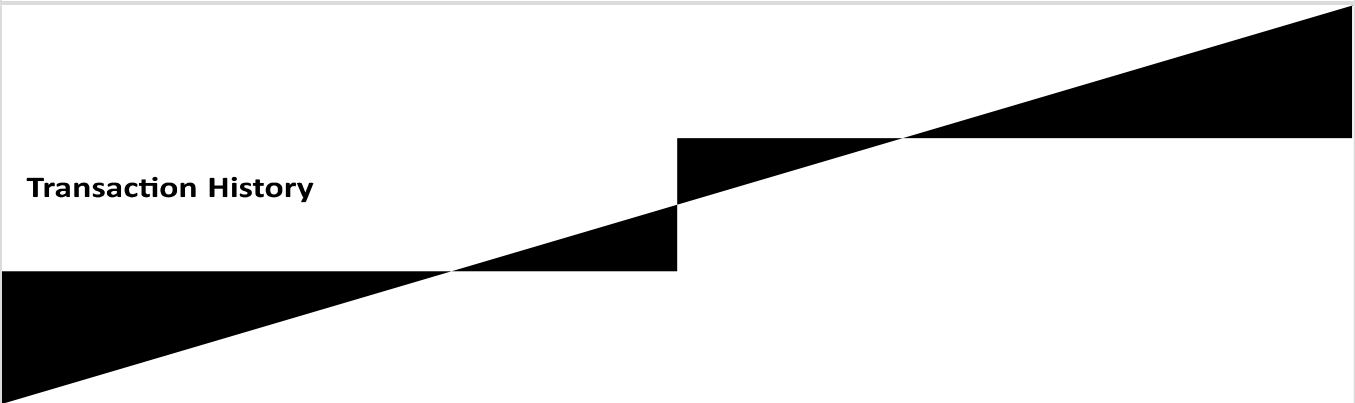
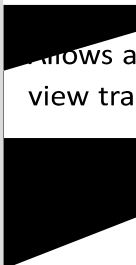
|                      |  |  |
|----------------------|--|--|
|                      |  |  |
| Registration Screen  |  |  |
| Customer Dashboard   |  |  |
| Deposit / Withdrawal |  |  |

Input field  
login and

Fields to  
password

Displays  
operation

Input field  
confirma

|  |   |
|--|---|
|  <p><b>Fund Transfer</b></p>        | <p>Enter ser<br/>amount.</p>  |
|  <p><b>Transaction History</b></p> | <p>Displays<br/>and amo</p>   |
| <p><b>Admin Dashboard</b></p>  |  <p>Shows a<br/>view tra</p> |
| <p><b>Reports Screen</b></p>   | <p>Admin ca<br/>transacti</p>   |

## 8. Project Applications and Limitations

### Applications

The **Automated Banking System (ABS)** has multiple applications in the banking and financial sector:

- **Banks and Financial Institutions:**

Automates customer account management, deposits, withdrawals, fund transfers, and transaction history.

- **Corporate Banking:**

Provides employees and clients with secure access to financial accounts and transactions.

- **Small Banks or Credit Unions:**

Enables digitization of banking operations, reducing paperwork and manual errors.

- **Customer Convenience:**

Provides 24/7 access to banking services, mini-statements, and fund transfers.

- **Administrative Use:**

Admins can monitor all transactions, generate reports, and manage accounts efficiently.

**Benefits:**

- Reduces dependency on manual operations.
- Increases speed and accuracy of banking operations.
- Improves customer satisfaction and transparency. Enhances
- security and reduces fraud risks.

**Limitations**

Despite its advantages, the system has some limitations:

- **Internet Dependency:** Requires stable internet for online operations.
- **Feature Limitations:** Advanced features like mobile banking, biometric authentication, and AI-based fraud detection are not included in this version.
- **Scalability:** Performance may decrease under extremely high user load without server optimization.
- **User Input Dependence:** Incorrect inputs can lead to errors in transactions.
- **Security Limitations:** Advanced encryption and multi-factor authentication are not implemented in the basic version.

## 9. Conclusion and Future Enhancement

**Conclusion**

The **Automated Banking System** provides a secure, reliable, and efficient platform to handle banking operations digitally. The system automates customer account management, transactions, and administrative tasks, reducing manual effort and errors.

By following the **Agile methodology**, the system was developed iteratively with continuous feedback and testing, ensuring each module works correctly before integration. The ABS is **user-friendly, secure, and scalable**, and allows customers to access their banking services 24/7 while enabling admins to monitor accounts and transactions effectively.

**Future Enhancements**

Future versions of the system may include:

- **Mobile Banking App:** Android/iOS application for better accessibility.
- **Biometric Authentication:** Fingerprint or facial recognition for secure login.
- **AI-based Fraud Detection:** Automatically detect suspicious transactions.
- **Multi-Currency Support:** Allow international transfers and foreign currency accounts.
- **Notifications & Alerts:** SMS/email alerts for transactions and account updates. **Enhanced**
- **Reporting:** Advanced analytics and dashboards for admins.

These enhancements will improve **security, convenience, and system efficiency**, making it a fully modern banking solution.



## 10. Bibliography & References

1. Pressman, R. S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 8th Edition.
2. Sommerville, I., *Software Engineering*, 10th Edition, Pearson Education.
3. IEEE Standard for Software Requirements Specification (SRS).
4. Oracle Java Documentation, <https://docs.oracle.com/javase/>
5. MySQL Official Documentation, <https://dev.mysql.com/doc/>
6. Agile Alliance – *Agile Methodology Guide*, <https://www.agilealliance.org>
7. TutorialsPoint – *Java, MySQL, and Banking System Tutorials*, <https://www.tutorialspoint.com>