Software Requirements Specification

For

Automated Banking System

Version 1.0 Approved

Prepared by : Shrasti Jadia

EnrollmentNo. : 0567CS221157

Submitted On: 05/01/2026

Submitted To: AMAR NAYAK

Software Engineering(Agile Methodology)

❖Table of Contents

1. Abstract

The **Automated Banking System (ABS)** is a secure and efficient software solution developed to modernize and simplify core banking operations. In traditional banking environments, many processes depend on manual work and paperwork, which often results in delays, transaction errors, and reduced operational efficiency. With the rapid expansion of digital banking services, there is a strong requirement for an automated system that delivers banking services with greater speed, accuracy, and reliability.

The ABS enables customers to perform essential banking operations such as **account creation, balance enquiry, cash deposit, withdrawal, fund transfer, and mini-statement generation** through a user-friendly interface. At the same time, administrators are provided with features to **manage customer accounts, monitor transaction activity, generate reports, and enforce security policies**, ensuring smooth functioning of the banking system.

The system is designed to minimize human intervention, reduce processing time, and improve service quality by providing **24×7 accessibility** and consistent performance. A major focus of the ABS is **security and data integrity**, ensuring that sensitive information such as account details, passwords, and transaction records remain confidential and protected against unauthorized access.

This project is developed using the **Agile Methodology**, where the software is built in **iterative sprints**. Each sprint delivers functional modules, supports continuous testing, and allows changes based on user and stakeholder feedback, resulting in a flexible and high-quality banking solution.

By implementing the Automated Banking System, banks can improve operational efficiency, reduce paperwork, enhance transaction accuracy, and provide better customer satisfaction. Future improvements may include **mobile banking integration, biometric authentication, real-time notification services, and AI-based fraud detection** to further strengthen digital banking capabilities.

2. Introduction

2.1 **Introduction**

The **Automated Banking System (ABS)** is a software-based application developed to computerize and simplify banking operations such as account management, customer services, financial transactions, and report generation. In traditional banking systems, many activities are performed manually, which often leads to delays, increased paperwork, human errors, and inefficiencies in maintaining customer records and transaction histories.

With the rapid growth of **digital banking and online financial services**, there is an increasing demand for a secure and reliable system that can automate routine banking tasks. The Automated Banking System provides a centralized platform where customers can perform essential operations such as **account creation, balance enquiry, deposit, withdrawal, fund transfer, and mini-statement generation**. At the same time, bank administrators can manage customer accounts, monitor transactions, maintain security controls, and generate reports for better decision-making.

2.2 Problem Identification

In many traditional banking environments, a large part of customer and transaction handling is still managed manually or through semi-automated systems. This leads to multiple operational challenges and affects both the bank staff and customers. Manual record maintenance increases processing time and creates a high chance of errors. In addition, customers often face long waiting times for basic services such as balance enquiry, withdrawals, deposits, and statement generation.

Some major issues observed in traditional banking systems include:

1. **Time-Consuming Manual Operations:**
   Banking tasks like account creation, deposits, withdrawals, and report preparation take more time due to manual processing.
2. **High Probability of Human Errors:**
   Manual data entry can lead to mistakes in transaction amounts, customer details, and account records, affecting reliability.

2.3 Need of the Project

The need for an **Automated Banking System (ABS)** arises due to the increasing demand for fast, accurate, and secure banking services. Modern banking requires automation to reduce workload, improve transaction speed, and provide better customer service. The Automated Banking System ensures that banking operations are performed digitally, reducing manual work and improving operational efficiency.

This project is required to:

1. **Automate Core Banking Operations:**
   Automate activities such as account creation, deposits, withdrawals, transfers, and statement generation to save time.
2. **Improve Accuracy and Reduce Errors:**
   Since transaction processing is computerized, the system reduces human mistakes and improves record correctness.
3. **Enhance Data Storage and Retrieval:**
   Store customer and transaction details in a centralized database for quick and reliable access.

The main objective of the **Automated Banking System (ABS)** is to provide a digital solution that manages banking operations efficiently while ensuring secure transactions and accurate record maintenance. The objectives are categorized as follows:

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.

2.5 Project Scheduling

Project scheduling is an important part of the Agile-based CRM system development. It defines the timeline, activities, and sprint-wise implementation plan to complete the project in an organized manner. In Agile methodology, the entire project is divided into small time-bound cycles called **Sprints**. Each sprint includes planning, design, coding, testing, and review.

Sprint No.	Sprint Name	Activities	Duration
1	Requirement Analysis	Requirement gathering, backlog preparation	1 Week
2	System Design	Database design, ER diagram, DFD, UI design	1 Week
3	Customer Module Development	Registration, login, balance enquiry	1 Week
4	Transaction Module	Deposit, withdrawal, fund transfer, mini-statement	2 Weeks
5	Admin Module	Manage accounts, monitor transactions, generate reports	1 Week
6	Testing & Deployment	Integration testing, bug fixing, deployment	1 Week

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 the Automated Banking System is to provide a computerized system that enables customers and bank staff to perform banking operations efficiently and securely. The system automates major services like account creation, deposit, withdrawal, balance inquiry, fund transfer, and transaction history. It reduces manual paperwork, saves time, and improves accuracy in banking processes.

3.2 Scope

The Automated Banking System is designed to manage essential banking operations in a digital manner. The system supports:

- Customer registration and login
- Account management
- Deposit and withdrawal transactions
- Money transfer between accounts
- Balance inquiry
- Transaction history display
- Admin/bank staff management features

This system can be used by banks to maintain customer records and enable fast and secure transactions.

3.3 Hardware Requirement / Software **Requirement**

Hardware Requirements

- Processor: Intel i3 / i5 or higher
- RAM: 4GB minimum (8GB recommended)
- Hard Disk: 50GB free space
- Monitor: 14” or above
- Keyboard & Mouse
- Internet connection (optional for online module)

Software Requirements

- Operating System: Windows 10/11
- Frontend: HTML, CSS, JavaScript (Optional)
- Backend: Java (JSP/Servlets)
- Database: MySQL
- Server: Apache Tomcat
- IDE: Eclipse / NetBeans
- Browser: Chrome / Firefox

3.4 Tools

Tools used for developing the Automated Banking System:

- Eclipse IDE / NetBeans
- Apache Tomcat Server
- MySQL Database
- MySQL Workbench / phpMyAdmin
- Git (optional)
- XAMPP (optional for database support)

3.5 Software Process Model

The project follows the **Waterfall Model**.
Steps involved:

1. Requirement Analysis
2. System Design
3. Implementation (Coding)
4. Testing
5. Deployment
6. Maintenance

The Waterfall model is selected because project requirements are clear and development is done in sequential phases. The Waterfall model consists of multiple phases such as **Requirement Analysis, System Design, Implementation (Coding), Testing, Deployment, and Maintenance**. In the first phase, requirements are collected from users and banking procedures are studied to identify system needs. In the design phase, overall architecture, database schema, ER diagram, and data flow diagrams are prepared. After finalizing the design, implementation is carried out using Java technologies (JSP/Servlets) and MySQL database. Once coding is completed, testing is performed to ensure correctness of functions such as secure login, correct balance update after deposit/withdraw, and accurate transaction recording.

This model also ensures better control over project progress because each stage has fixed deliverables like SRS, design documents, source code, and test cases. Any errors found during testing are corrected before final deployment. Finally, in the maintenance phase, the system is updated to fix bugs and to include new banking features as per future requirements.

4. System Design

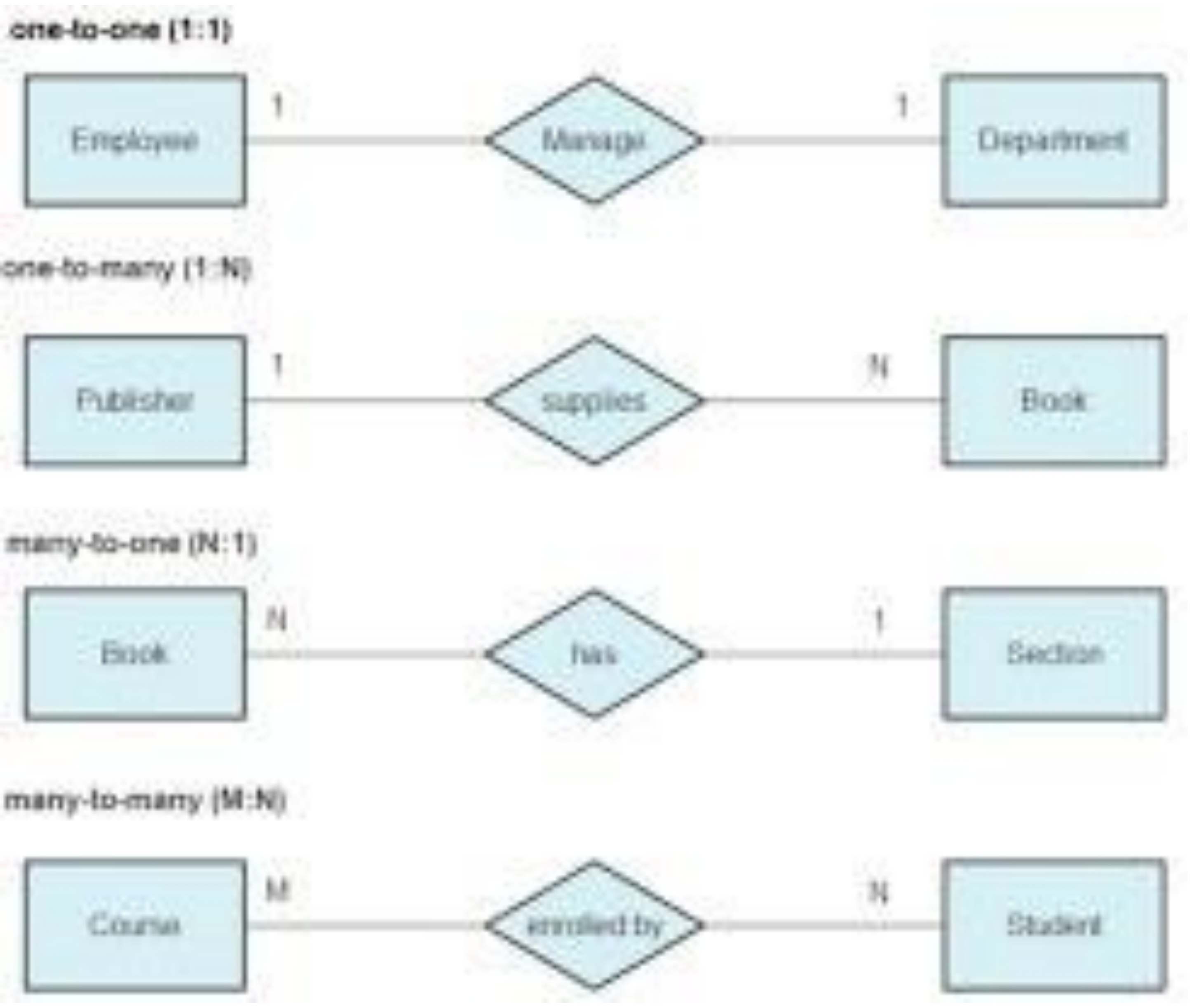
4.1 Data Dictionary

Data Dictionary is an important document used in the **System Design phase** of software development. It is a structured collection of information that describes the **data elements** used in a system. It provides complete details about the database such as **tables, fields (attributes), data types, size, constraints, and description** of each field.

Entity	**Attribute**	**Description**
Customer	customer_id	Unique ID for each customer
	name	Full name of customer
	email	Email used for login and communication
	password	Login password for security
	account_no	Bank account number
	balance	Current account balance
Account	account_no	Unique account number
	customer_id	ID of the account holder
	account_type	Savings / Current / Other
Transaction	transaction_id	Unique ID of each transaction
	account_no	Account involved in transaction
	type	Deposit / Withdrawal / Fund Transfer
	amount	Amount of transaction
	date	Date of transaction
Admin	admin_id	Unique ID of administrator
	name	Name of admin

4.2 ER Diagram (Entity-Relationship Diagram)

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



Entities:

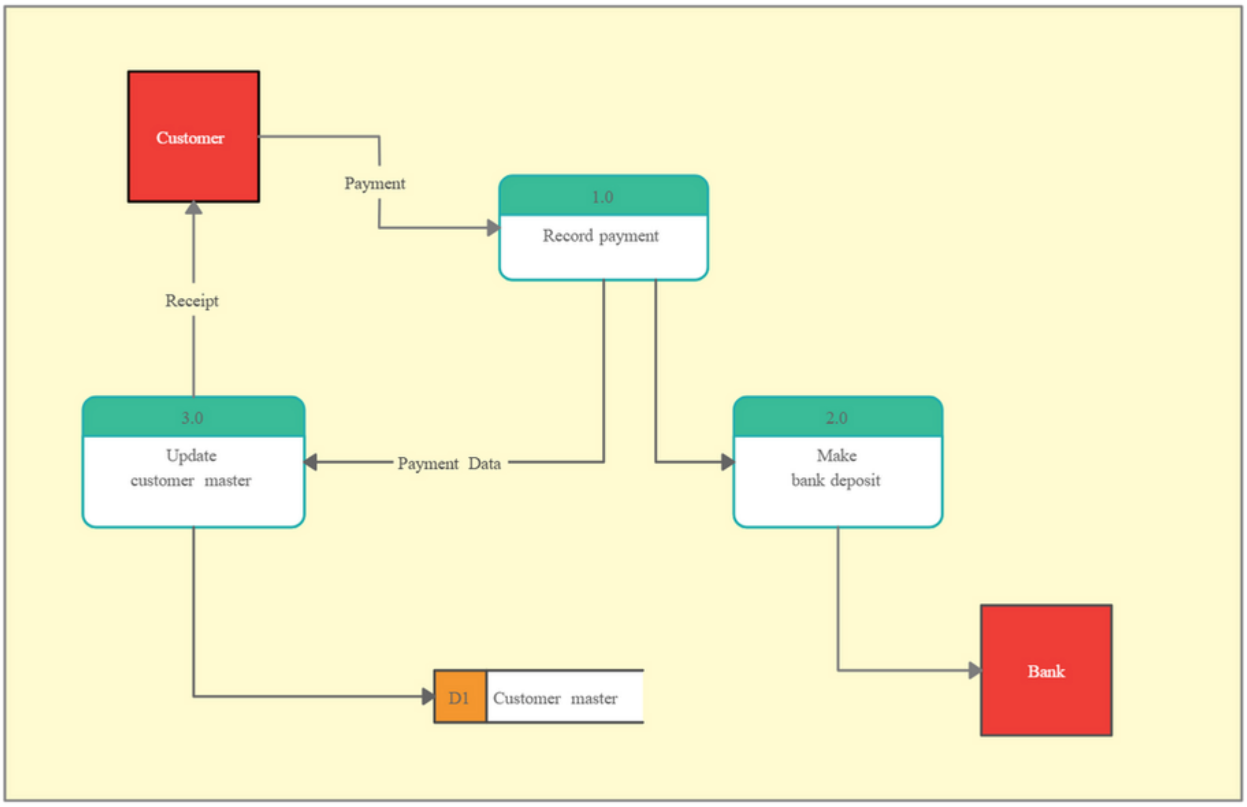
- **Customer**
- **Account**
- **Transaction**
- **Admin**

Relationships:

- One customer can have **one or more accounts**
- One account can have **multiple transactions**
- Admin manages customers and accounts

4.3 Data Flow Diagram (DFD)

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



DFD Level 0

Main entities:

- Customer
- Admin
- Automated Banking System

Processes:

- Login/Registration
- Account handling
- Transaction processing
- Report generation

DFD Level 1

Modules:

1. **Authentication Module**
2. **Account Management Module**
3. **Transaction Module**

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);

   }


 publicvoid 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:

Screens included in the system:

1. Home Page
2. Customer Registration Page
3. Login Page
4. Customer Dashboard
5. Deposit Screen
6. Withdraw Screen
7. Transfer Money Screen
8. Balance Inquiry Screen
9. Transaction History Screen
10. Admin Dashboard

Screen	Description
Login Screen	Input email and password for customers or admin. Provides secure authentication.
Registration Screen	New users provide details to create a bank account.
Customer Dashboard	Displays account info, balance, and available operations (deposit, withdraw, transfer).
Deposit / Withdrawal	Input amount to deposit or withdraw. Displays confirmation and updated balance.
Fund Transfer	Transfer money between accounts with confirmation and updated balances.
Transaction History / Mini-Statement	Displays last few transactions for the customer account.
Admin Dashboard	Allows admins to manage users, view transactions, and generate reports.

6. Testing

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

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

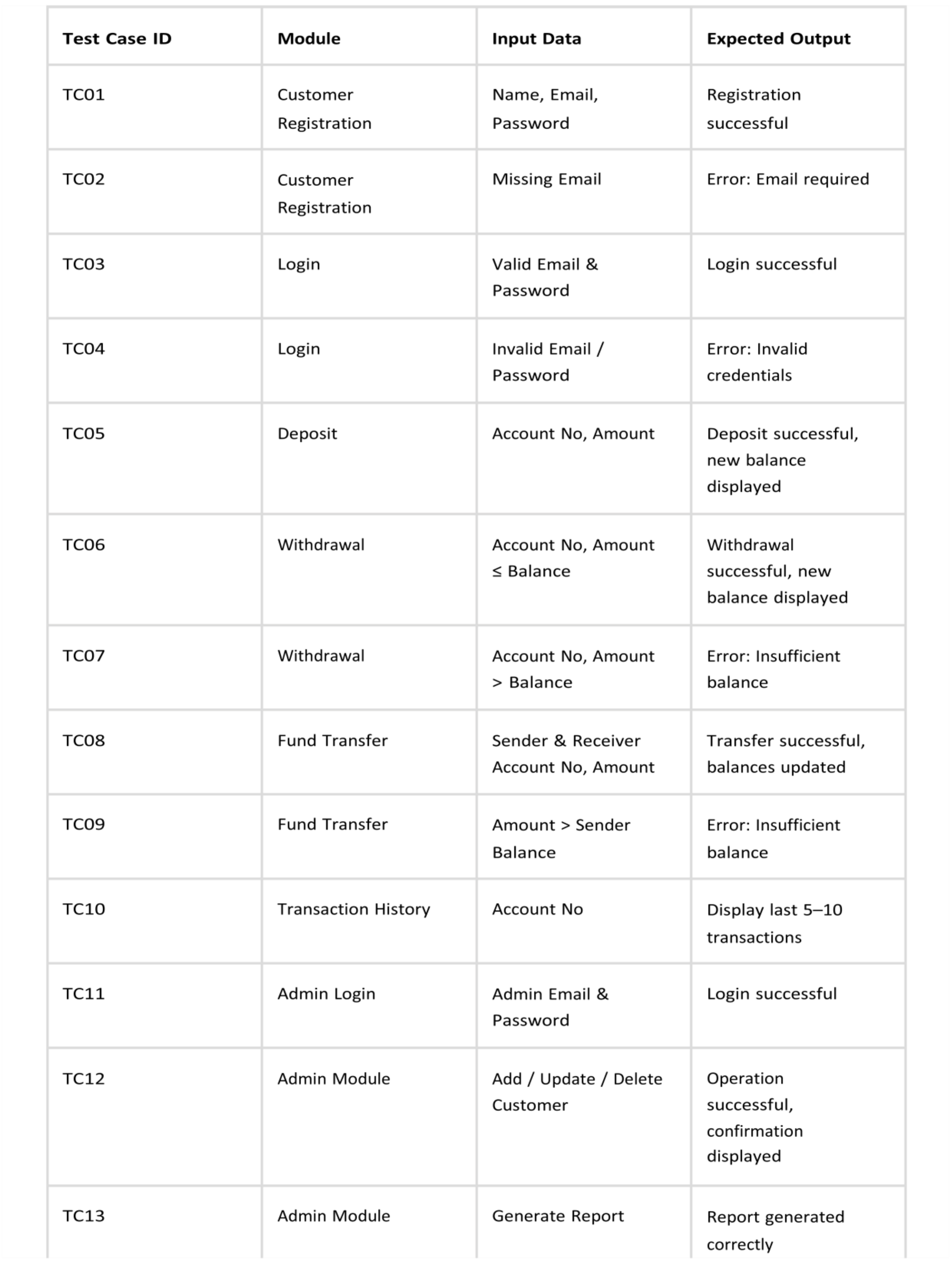
6.1 Test Data

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

- Login: correct and incorrect credentials
- Deposit: 1000, 5000
- Withdraw: 100, 2000
- Transfer: valid and invalid account number
- Balance check after each transaction

6.2 Test Result

Test Case	**Input**	**Expected Output**	**Result**
Login	valid username/password	login success	Pass
Login	wrong password	error message	Pass
Deposit	amount=1000	balance increased	Pass
Withdraw	amount > balance	insufficient balance	Pass
Transfer	invalid account	invalid account error	Pass



Test Case ID	**Module**	**Input Data**	**Expected Output**
TC01	Customer Registration	Name, Email, Password	Registration successful
TC02	Customer Registration	Missing Email	Error: Email required
TC03	Login	Valid Email & Password	Login successful
TC04	Login	Invalid Email / Password	Error: Invalid credentials
TC05	Deposit	Account No, Amount	Deposit successful, new balance displayed
TC06	Withdrawal	Account No, Amount ≤ Balance	Withdrawal successful, new balance displayed
TC07	Withdrawal	Account No, Amount > Balance	Error: Insufficient balance
TC08	Fund Transfer	Sender & Receiver Account No, Amount	Transfer successful, balances updated
TC09	Fund Transfer	Amount > Sender Balance	Error: Insufficient balance
TC10	Transaction History	Account No	Display last 5–10 transactions
TC11	Admin Login	Admin Email & Password	Login successful
TC12	Admin Module	Add / Update / Delete Customer	Operation successful, confirmation displayed
TC13	Admin Module	Generate Report	Report generated correctly

7. User Manual

7.1 How to Use Project Guidelines

Steps to run the project:

1. Install MySQL and create database
2. Import database tables
3. Open project in Eclipse
4. Configure Tomcat server
5. Run project on server
6. Open in browser using localhost link

7.2 Screen Layouts and Description

- **Login Screen:** customer enters credentials
- **Dashboard:** options like deposit, withdraw, transfer, view balance
- **Transaction Page:** shows date, type, amount
- **Admin Panel:** manage customers, view all accounts

8. Project Applications and Limitations

The **Automated Banking System** has a wide range of real-world applications in the banking sector. It is mainly used to **automate banking transactions** such as deposit, withdrawal, balance inquiry, and fund transfer, which helps banks provide faster and more efficient services to customers. This system reduces the burden of manual record keeping and paperwork, thereby minimizing human errors and improving accuracy. It also provides **quick access to customer and account details**, enabling bank staff to manage customer records easily and respond to customer requests efficiently. Furthermore, the system ensures **secure digital storage of banking data** and maintains transaction records in an organized manner, which helps in auditing and report generation.

However, the system also has some limitations. The Automated Banking System works effectively only when there is a **stable server and proper network connectivity**, as banking operations depend on continuous system access. It is designed for basic banking operations and may not support large-scale functionalities used in **core banking systems** that handle millions of users and complex services. The system also requires strong **database security, regular backups, and access control** to prevent unauthorized access and data loss. Additionally, since the project is limited to essential features, advanced facilities like loan processing, credit card services, OTP verification, and real-time interbank transactions are not included.

9. Conclusion and Future Enhancement

Conclusion

The **Automated Banking System** successfully provides a reliable and efficient solution for managing basic banking operations in a computerized manner. The system automates important services such as customer registration, secure login, deposit, withdrawal, balance enquiry, fund transfer, and transaction history management. This automation reduces manual paperwork and minimizes chances of human errors, making banking services faster and more accurate. The system also improves customer experience by providing quick access to account information and transaction details. Additionally, it ensures secure and well-organized digital record maintenance, which helps banks in tracking transactions, generating reports, and maintaining transparency. Overall, the Automated Banking System is a user-friendly and effective application that fulfills the required objectives and provides a strong base for further development into a complete banking solution.

Future Enhancements

- Add OTP-based transaction verification
- Add ATM simulation module
- Add loan management module
- Add mobile app support
- Generate monthly statement PDF
- Add advanced security (encryption, 2FA)

10. Bibliography and References

1. Java Documentation – Oracle
2. MySQL Documentation
3. Apache Tomcat Documentation
4. JDBC API Documentation
5. Software Engineering Book by Roger Pressman
6. Internet sources and tutorials (GeeksforGeeks, Javatpoint)

Report Finalized and Submitted by: Shrasti Jadia.

Submission Date: January 5, 2026.

,