

Darshan University

A Project Report on

**“Bank Management System”**

Under the subject

**Software Engineering (2301CS405)**

B. Tech, Semester – IV

Computer Science & Engineering Department

|  |  |
| --- | --- |
| Submitted By | |
| Student Name: Utsav Boghani | Enrollment No.: 23010101037 |
| Academic Year  (2024-2025) | |
| Internal Guide  Prof. R. B. Gondaliya  Darshan University | Dean-DIET  Dr. Gopi Sanghani  Darshan University |

|  |  |
| --- | --- |
|  | **Computer Science & Engineering Department**  **Darshan University** |

**DECLARATION**

We hereby declare that the SRS, submitted along with the **Software Engineering** **(2301CS405)** for entitled **“Bank Management System”** submitted in partial fulfilment for the Semester-4 of **Bachelor Technology (B. Tech)** in **Computer Science and Engineering (CSE)** Departmentto Darshan University, Rajkot, is a record of the work carried out at **Darshan University, Rajkot** under the supervision of R. B. Gondaliya and that no part of any of report has been directly copied from any students’ reports, without providing due reference.

(Utsav Boghani)

Student’s Signature

Date: \_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  | **Computer Science & Engineering Department**  **Darshan University** |

**CERTIFICATE**

This is to certify that the SRS on **“Bank Management System” has** been satisfactorily prepared by **Utsav Boghani** (**23010101037**) under my guidance in the fulfillment of the course **Software Engineering (2301CS405)** work during the academic year 2024-2025.

|  |  |  |
| --- | --- | --- |
| Internal Guide  Prof. R. B. Gondaliya  Darshan University |  | Dean-DIET  Dr. Gopi Sanghani  Darshan University |

**Acknowledgement**

I wish to express my sincere gratitude to my project guide Prof. R. B. Gondaliyaand all the faculty members for helping me through my project by giving me the necessary suggestions and advices along with their valuable co- ordination in completing this work.

I also thank my parents, friends and all the members of the family for their precious support and encouragement which they had provided in completion of my work. In addition to that, I would also like to mention the Darshan University personals who gave me the permission to use and experience the valuable resources required for the project from the University premises.

Thus, in conclusion to the above said, I once again thank the faculties and members of **Darshan University** for their valuable support in completion of the project.

Thanking You

**Utsav Boghani**

**ABSTRACT**

**The Bank Management System (BMS) is a sophisticated software application crafted to streamline banking operations with exceptional efficiency. It amalgamates core banking functionalities, offering seamless access to services for both users and administrators. BMS ingeniously abstracts intricate financial operations, rendering them through a user-friendly interface that caters to various stakeholders, including account holders and bank employees. Through the BMS, users can perform essential tasks such as money transfers, balance inquiries, and account management, while administrators can efficiently oversee operations like interest calculations, customer database maintenance, and transaction monitoring. One of the key strengths of the BMS lies in its foundation on object-oriented programming principles, with a particular focus on abstraction. This principle plays a crucial role in concealing the complexities of implementation from the end-users. For example, when a customer views their account balance, they interact with a simple interface that masks the underlying processes involving complex database queries and calculations. Similarly, transactions and fund transfers, although powered by sophisticated algorithms, are presented as straightforward interactions on the platform.**

**This approach to abstraction is not only about ease of use but also significantly enhances security. Users can confidently interact with the system, knowing their sensitive data is securely processed without any exposure to the intricate underlying mechanisms. Abstraction in BMS ensures that users experience a simplified, efficient, and secure environment for managing their financial activities. Additionally, the BMS leverages the concept of modularity to enhance its functionality. Modularity involves compartmentalizing the system into distinct components or modules. Each module operates independently, which allows developers to update or enhance specific features without causing disruptions to the entire system. This compartmentalization facilitates easier maintenance and scalability of the software. For instance, a module responsible for handling transactions can be updated to incorporate new features or improve performance without affecting modules that manage other functionalities like customer records or interest calculations. By isolating each function within its module, BMS ensures that any changes or enhancements are localized, thereby reducing the risk of errors or system downtime. This design philosophy makes the BMS not only robust but also flexible and adaptable to future needs and technological advancements. In summary, the Bank Management System (BMS) encapsulates the intricate details of banking operations into an intuitive and user-friendly interface. This abstraction layer benefits both users and developers by ensuring efficient, secure, and user-friendly financial management. Users enjoy a seamless experience where complex operations are simplified, and their data is kept secure. Meanwhile, developers benefit from a modular structure that facilitates easy updates and maintenance, ensuring the system remains up-to-date and efficient without significant downtime. The success of a BMS is measured by its ability to provide reliable, secure, and efficient banking operations. Its design principles, grounded in abstraction and modularity, contribute significantly to achieving these goals. For users, it means an effortless way to manage their financial affairs, while for banks, it translates into an operationally efficient system that can adapt to the evolving needs of the financial sector. As technology continues to evolve, the principles embedded in the design of BMS ensure that it remains a critical tool in the efficient management of banking operations, delivering value to both users and administrators alike. In essence, the BMS is a testament to the power of software engineering principles in creating systems that are both powerful and user centric. By abstracting the complexities and employing a modular design, it provides a platform that is not only effective in its current form but is also poised for future enhancements and scalability, making it an indispensable component of modern banking infrastructure.**

**Table of Contents**

[List of Figures I](#_Toc189471925)

[List of Tables II](#_Toc189471926)

[1 Introduction: 1](#_Toc189471927)

[1.1 Functional Requirement 4](#_Toc189471928)

[1.2 1. For Customers: 4](#_Toc189471929)

[Customers are the end-users who use the banking system for personal banking needs. Their functionalities include: 4](#_Toc189471930)

[1.3 2.For Bank Employees: 5](#_Toc189471931)

[Bank employees manage and support banking operations and customer services. Their functionalities include: 5](#_Toc189471932)

[1.4 3.For Administrators: 5](#_Toc189471933)

[Administrators oversee system operations, manage user roles, and ensure compliance. Their functionalities include: 5](#_Toc189471934)

[1.5 4.For Loan Officers: 6](#_Toc189471935)

[Loan officers specialize in handling loan-related services and customer interactions. Their functionalities include: 6](#_Toc189471936)

[1.6 Non-Functional Requirement 6](#_Toc189471937)

[2 Design and Implementation Constraints 7](#_Toc189471938)

[2.1 Activity diagram and Swimlane diagram 8](#_Toc189471939)

[2.2 Sequence diagram 11](#_Toc189471940)

[2.3 State diagram 12](#_Toc189471941)

[2.4 Class diagram 13](#_Toc189471942)

[2.5 Data flow diagram 14](#_Toc189471943)

[2.5.1 Context diagram (level-0) 14](#_Toc189471944)

[2.5.2 DFD Level-1 15](#_Toc189471945)

[3 External interface requirement (Screens) 16](#_Toc189471946)

[3.1 Screen-1: Registration New Account 16](#_Toc189471947)

[3.2 Screen-2: Deposit Money Form 17](#_Toc189471948)

[3.3 Screen-3: Withdraw Money 18](#_Toc189471949)

[3.4 Screen-3: Account Statement 19](#_Toc189471950)

[3.5 Screen-3: Fund Transfer 20](#_Toc189471951)

[3.6 Screen-3: Loan Services 21](#_Toc189471952)

[3.7 Screen-3: Card Services 22](#_Toc189471953)

[3.8 Screen-3: Fixed Deposit Services 23](#_Toc189471954)

[4 Database design 24](#_Toc189471955)

[4.1 List of Tables 24](#_Toc189471956)

[5 Stories and Scenario 26](#_Toc189471957)

[5.1 Story-1: Open a New Bank Account 26](#_Toc189471958)

[5.1.1 Scenario# S1.1 26](#_Toc189471959)

[5.1.2 Scenario# S1.2 26](#_Toc189471960)

[5.2 Story-2: Transfer Funds Between Accounts 27](#_Toc189471961)

[5.2.1 Scenario# S1.1 27](#_Toc189471962)

[5.2.2 Scenario# S1.2 27](#_Toc189471963)

[5.3 Story-3: Generate Account Statements 28](#_Toc189471964)

[5.3.1 Scenario# S1.1 28](#_Toc189471965)

[5.3.2 Scenario# S1.2 28](#_Toc189471966)

[6 Test cases 29](#_Toc189471967)

[7 References 33](#_Toc189471968)

# List of Figures

[Figure 2.1‑1 Activity diagram for Loan Services 8](#_Toc189471969)

[Figure 2.2‑1 Sequence diagram for ATM 11](#_Toc189471970)

[Figure 2.3‑1 State diagram of ATM 12](#_Toc189471971)

[Figure 2.4‑1 Class diagram for Bank management system 13](#_Toc189471972)

[Figure 2.5‑1 Context diagram for Bank management system 14](#_Toc189471973)

[Figure 2.5‑2 DFD level-1 for Bank management system 15](#_Toc189471974)

[Figure 3.1‑1 Screen-1: Registration New Account 16](#_Toc189471975)

[Figure 3.2‑1 Screen-2: Deposit Money Form 17](#_Toc189471976)

[Figure 3.3‑1 Screen-3: Withdraw Money 18](#_Toc189471977)

[Figure 3.4‑1 Screen-3: Account Statement 19](#_Toc189471978)

[Figure 3.5‑1 Screen-3: Fund Transfer 20](#_Toc189471979)

[Figure 3.5‑1 Screen-3: Loan Services 21](#_Toc189471980)

[Figure 3.5‑1 Screen-3: Card Services 22](#_Toc189471981)

[Figure 3.5‑1 Screen-3: Fixed Deposit Services 23](#_Toc189471982)

# List of Tables

[Table 3.1‑1 Screen element of Registration New Account 16](#_Toc189471996)

[Table 3.2‑1 Screen element of Deposit Money Form 17](#_Toc189471997)

[Table 3.3‑1 Screen element of Withdraw Money 18](#_Toc189471998)

[Table 3.4‑1 Screen element of Account Statement 19](#_Toc189471999)

[Table 3.5‑1 Screen element of Fund Transfer 20](#_Toc189472000)

[Table 3.5‑1 Screen element of Loan Services 21](#_Toc189472001)

[Table 3.5‑1 Screen element of Card Services 22](#_Toc189472002)

[Table 3.5‑1 Screen element of Fixed Deposit Services 23](#_Toc189472003)

[Table 4.1‑1 Table: Customers 24](#_Toc189472004)

[Table 4.1‑2 Table: Accounts 24](#_Toc189472005)

[Table 4.1‑3 Table: Transactions 24](#_Toc189472006)

[Table 4.1‑4 Table: Employees 25](#_Toc189472007)

[Table 4.1‑4 Table: Loans 25](#_Toc189472008)

# Introduction:

A Bank Management System (BMS) is an integral part of modern banking infrastructure, designed to handle the complexities of financial transactions, customer data management, and regulatory compliance. It provides a centralized platform for banks to manage their operations, enhancing efficiency, security, and customer satisfaction. In today's fast-paced financial environment, the BMS is indispensable. It allows customers to access banking services through various channels such as ATMs, online platforms, and mobile apps, while helping banks streamline their internal processes, such as ledger maintenance, account reconciliation, and loan processing. By offering a centralized system, the BMS ensures that all financial operations are handled smoothly and efficiently, thereby improving the overall banking experience for both customers and employees.

The BMS is designed to be comprehensive, encompassing multiple modules that address a variety of functionalities. One of the primary modules is Customer Management, which handles customer details, account types, and the services provided to them. This module ensures that customer data is accurately maintained and easily accessible, facilitating efficient service delivery. Another critical module is Transaction Handling, which records various types of transactions, such as deposits, withdrawals, and transfers. This module ensures that all transactions are processed smoothly and recorded accurately, maintaining the integrity of financial records.

Loan Management is another essential module within the BMS, facilitating loan applications, approvals, and tracking repayments. This module ensures that both the bank and customers can monitor loan statuses effectively, making the loan management process more transparent and efficient. Additionally, the BMS incorporates robust Security Features that implement strong authentication mechanisms, data encryption, and fraud detection protocols to protect both customers and the bank. With increasing cyber threats, these security measures are crucial in safeguarding sensitive financial information and preventing unauthorized access.

Regulatory Compliance is a significant aspect of the BMS, ensuring that the system adheres to government and financial regulations. This module provides a safe and legal framework for banking operations, protecting the bank from potential legal issues and building trust with customers by assuring them that their data and transactions are secure. The implementation of a Bank Management System brings numerous benefits to both banks and customers. Enhanced Customer Experience is one such benefit, allowing customers to enjoy round-the-clock access to banking services, ensuring convenience and satisfaction. With online and mobile banking platforms, customers can manage their accounts from anywhere at any time, providing unparalleled flexibility and accessibility.

Operational Efficiency is significantly improved with the implementation of a BMS. Automation reduces the need for manual tasks, leading to fewer errors and more streamlined processes. This efficiency allows employees to focus on more strategic tasks rather than routine administrative work, enhancing overall productivity. Data Security is another critical benefit, as sensitive financial data is protected through advanced security protocols, safeguarding customer information and preventing potential fraud. Scalability is also a vital feature of the BMS, ensuring that the system can grow in line with the increasing demands of the bank and its customer base. The modular design allows for easy updates and enhancements without disrupting the entire system, making it adaptable to future needs.

Object-oriented programming principles, particularly abstraction, play a crucial role in the BMS. Abstraction hides the complexities of implementation from the end-users, ensuring a smooth and intuitive experience. For example, when a customer views their account balance, they interact with a simple interface that masks the underlying processes involving complex database queries and calculations. Similarly, transactions and fund transfers, although powered by sophisticated algorithms, are presented as straightforward interactions on the platform. This abstraction ensures that users can confidently interact with the system without needing to understand the underlying technical details.

Modularity is another fundamental principle in the design of the BMS. By compartmentalizing functions into distinct modules, the system becomes more manageable and scalable. Developers can update or enhance specific features without disrupting the entire system, facilitating easier maintenance and scalability. For instance, if the loan management module needs an update, it can be done independently without affecting the transaction handling or customer management modules. This modular approach ensures that the BMS can adapt to the bank's evolving needs and technological advancements.

Enhancing security and ensuring regulatory compliance are critical aspects of the BMS. Strong authentication, data encryption, and fraud detection measures protect both customers and the bank from cyber threats. Regulatory compliance ensures that the system adheres to various government and financial regulations, providing a safe and legal framework for banking operations. Compliance with regulations not only protects the bank from legal issues but also builds trust with customers, assuring them that their data and transactions are secure.

The Bank Management System encapsulates the intricate details of banking operations into an intuitive and user-friendly interface. This abstraction benefits both users and developers, ensuring efficient, secure, and user-friendly financial management. Users experience a seamless and convenient way to manage their financial activities, while developers benefit from a system that is easy to maintain and scalable. In essence, the BMS is a cornerstone of modern banking, driving innovation and operational excellence. It enables banks to offer better services to customers, streamline their internal processes, and stay compliant with regulations. As technology continues to evolve, the principles embedded in the design of the BMS ensure that it remains a critical tool in the efficient management of banking operations, delivering value to both users and administrators alike.

Looking to the future, Bank Management Systems will continue to evolve and adapt to new challenges and opportunities. With advancements in technology such as artificial intelligence and blockchain, BMS can become even more efficient and secure. AI can help in predictive analysis and personalized customer services, while blockchain can enhance transaction security and transparency. The integration of these advanced technologies will further improve the functionalities of BMS, making banking operations even more efficient and secure. Additionally, as customer expectations continue to rise, BMS will need to offer more innovative and personalized services to meet these demands.

In conclusion, a Bank Management System (BMS) is an essential component of modern banking infrastructure. It provides a centralized platform that enhances the efficiency, security, and customer satisfaction of banking operations. By leveraging object-oriented programming principles and modularity, BMS ensures that complex financial operations are simplified and securely managed. The benefits of implementing a BMS include enhanced customer experience, improved operational efficiency, robust data security, and scalability. As technology advances, the role of BMS will become even more critical in driving innovation and operational excellence in the banking sector. The future holds immense potential for BMS to evolve and adapt, ensuring that it continues to be a cornerstone of modern banking.

## Functional Requirement

## 1. For Customers:

## Customers are the end-users who use the banking system for personal banking needs. Their functionalities include:

1. **Account Services**
   * Open new accounts, view account details, and close accounts.
2. **Transaction Processing**
   * Perform deposits, withdrawals, and fund transfers between accounts.
3. **Statement Generation**
   * Request detailed account statements for specific time periods.
4. **Loan Services**
   * Apply for loans and track approvals and repayment schedules.
5. **Password Handling**
   * Reset passwords securely through multi-step verification.
6. **ATM Locator**
   * Display nearby ATMs based on the user’s current location.
7. **Notifications**
   * Send SMS or email notifications for account activities and updates.
8. **Transaction History**
   * View a detailed history of all account transactions.
9. **Currency Exchange**
   * Perform currency conversions and display exchange rates.
10. **Mobile Banking**
    * Provide a mobile application for seamless access to banking services.
11. **Account Alerts**
    * Notify users of low balance, overdrafts, or unusual account activities.
12. **Nominee Details**
    * Add or update nominee information for accounts.
13. **Branch Locator**
    * Display nearby branches based on user location.
14. **Multi-Currency Accounts**
    * Provide support for maintaining accounts in multiple currencies.
15. **Joint Account Services**
    * Facilitate co-owned accounts with specific access permissions.
16. **Transaction Limits**
    * Enable setting daily or monthly transaction limits.
17. **Promotional Offers**
    * Display special offers and discounts on services.
18. **Budget Planning**
    * Offer tools for planning and tracking budgets.
19. **Digital Vault**
    * Provide secure storage for important digital documents.
20. **Tax Certificates**
    * Allow users to generate and download tax-related documents.
21. **Bill Payment**
    * Enable payment of utility and credit card bills.
22. **Cheque Book Requests**
    * Allow users to place or cancel cheque book orders.
23. **Live Currency Updates**
    * Display real-time currency exchange rates.
24. **Fund Investments**
    * Enable investments in mutual funds and fixed-income products.
25. **Interest Rate Alerts**
    * Notify users about changes in applicable interest rates.
26. **Cardless Cash Withdrawals**
    * Allow withdrawals via mobile app without a physical card.
27. **Expense Categorization**
    * Automatically categorize transactions for financial tracking.
28. **Loan Calculator**
    * Estimate loan EMIs using an integrated calculator.
29. **Cross-Bank Transactions**
    * Facilitate payments and transfers to other bank accounts.
30. **Profile Updates**
    * Allow users to update personal details like address and contact information.

## 2.For Bank Employees:

## Bank employees manage and support banking operations and customer services. Their functionalities include:

1. **Fixed Deposit Services**
   * Create and track fixed deposit accounts.
2. **Customer Support**
   * Provide integrated systems for handling user queries through chat or messaging platforms.
3. **Customer Feedback Collection**
   * Collect and process user feedback for service enhancement.
4. **Cheque Book Services**
   * Process cheque book orders or cancellations.
5. **Audit Trail**
   * Maintain detailed records of all system activities for compliance.
6. **Role-Based Access**
   * Implement permissions based on the user role.
7. **Secure Messaging**
   * Enable encrypted communication between users and the bank.
8. **Promotional Offers**
   * Display promotional services and discounts to customers.

## 3.For Administrators:

## Administrators oversee system operations, manage user roles, and ensure compliance. Their functionalities include:

1. **User Authentication**
   * Provide secure login mechanisms for all users.
2. **Security Enhancements**
   * Implement data encryption and fraud detection protocols.
3. **Data Backup**
   * Conduct regular backups to prevent data loss.
4. **Activity Logs**
   * Maintain detailed logs for auditing and monitoring.
5. **Interest Rate Tools**
   * Include tools for calculating interest rates on savings, loans, and deposits.
6. **Card Services**
   * Facilitate activation, blocking, and replacement of debit and credit cards.
7. **Overdraft Facilities**
   * Provide overdraft services for eligible users.
8. **Biometric Features**
   * Support biometric authentication, including fingerprint and facial recognition.
9. **Transaction Limits**
   * Allow users to define transaction limits.
10. **Fraud Detection**
    * Monitor for unusual activities and send alerts.
11. **Customizable Dashboard**
    * Provide administrators with an interface for system insights.
12. **Cross-Bank Transactions**
    * Enable transactions across different banking networks.
13. **Expense Categorization**
    * Provide tools for tracking and categorizing expenses.

## 4.For Loan Officers:

## Loan officers specialize in handling loan-related services and customer interactions. Their functionalities include:

1. **Loan Services**
   * Handle customer loan applications and repayment tracking.
2. **Loan Eligibility Details**
   * Display eligibility criteria and guidelines for applying for loans.
3. **Loan Calculator**
   * Estimate EMIs based on customer inputs.
4. **Interest Rate Notifications**
   * Notify users of interest rate changes on loans.
5. **Spending Analysis**
   * Use analytics tools to assess spending patterns and repayment capacities.

## Non-Functional Requirement

**1**. **Scalability:** The system should handle an increasing number of users and transactions.

**2**. **Performance:** All transactions should be processed within a maximum of 2 seconds.

**3**. **Reliability:** The system must ensure 99.9% uptime to avoid disruptions.

**4**. **Security:** Data should be encrypted during storage and transmission.

**5**. **Usability:** The interface should be intuitive and user-friendly for customers and staff.

**6**. **Maintainability:** The system should allow easy updates and integration of new features.

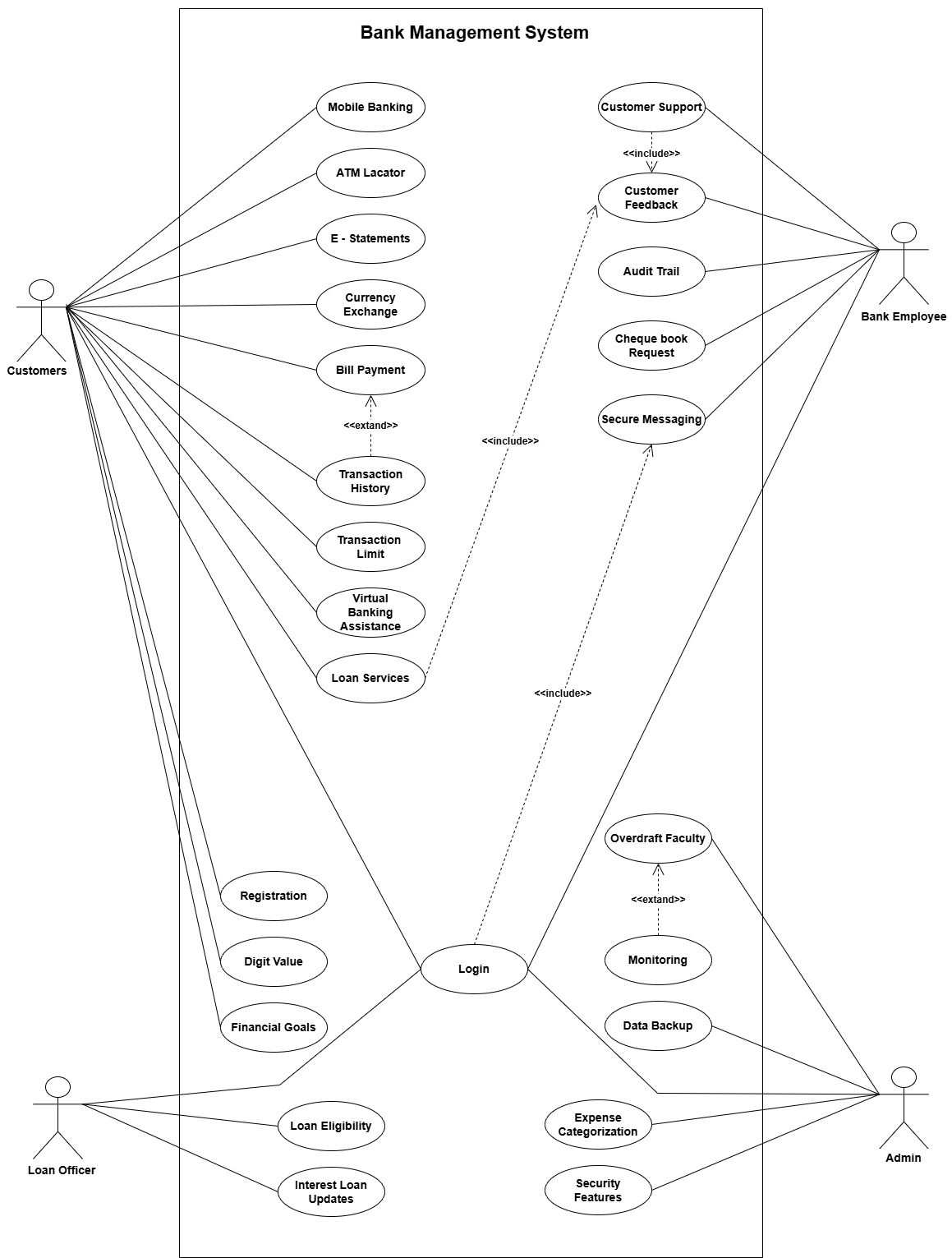
**7**. **Portability:** The system must support multiple platforms, including mobile and desktop devices.

**8**. **Compliance:** Adherence to financial regulations such as GDPR and PCI DSS.

**9**. **Data** **Backup:** Regular backups should be maintained to prevent data loss.

**10**. **Logging** **and** **Monitoring:** The system must provide detailed logs for auditing purposes.

# Design and Implementation Constraints

­­­ Use case diagram for Bank Management System

## Activity diagram and Swimlane diagram

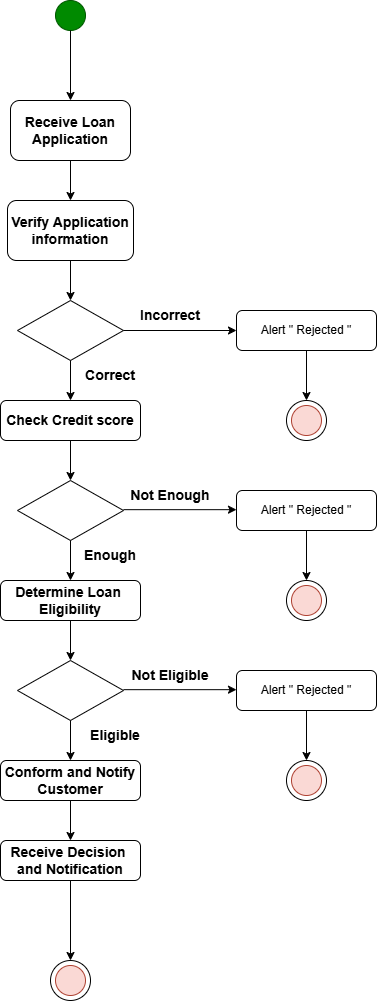
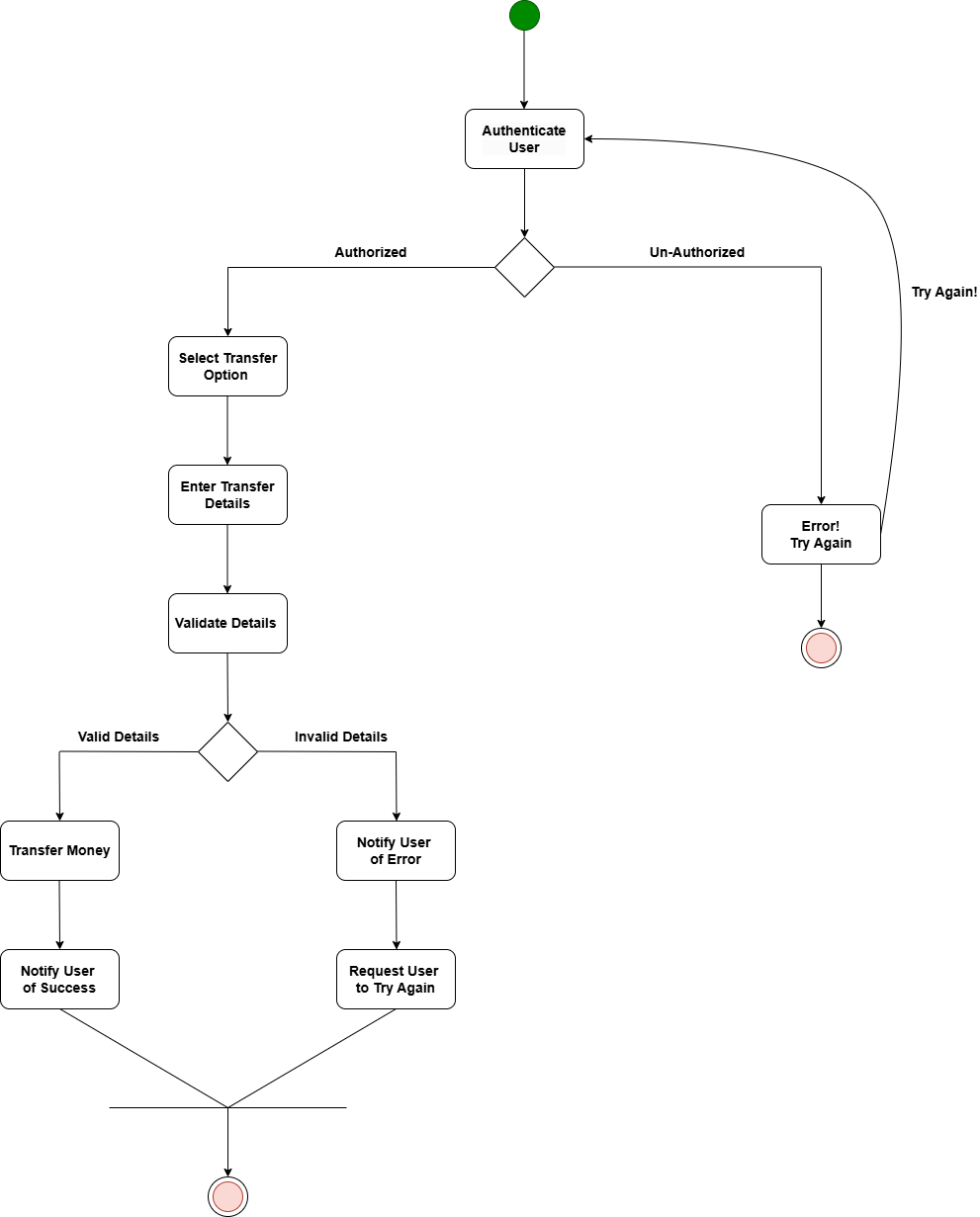


Figure 2.1‑1 Activity diagram for Loan Services



Money Transfer

Figure 2.1‑2 Activity diagram for Money Transfer

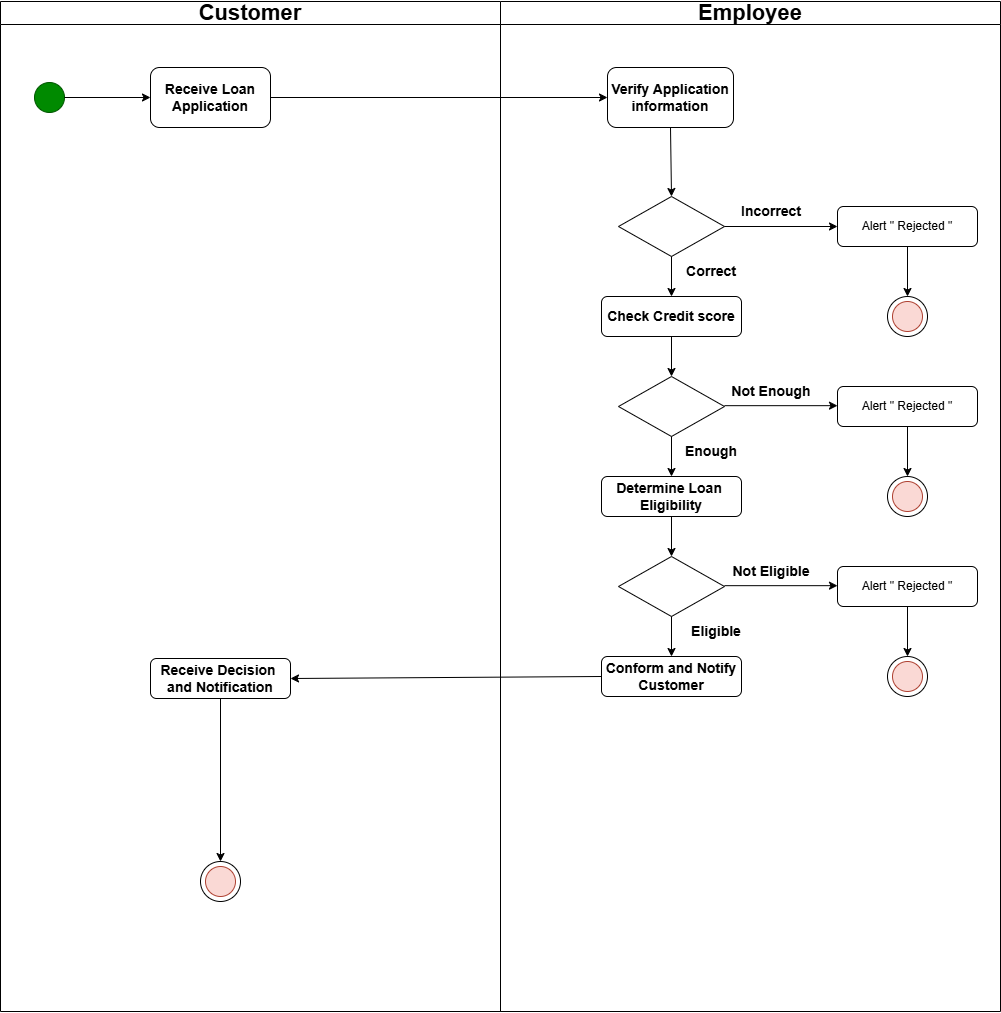


Figure 2.1‑3 Swimlane diagram for Loan Services

## Sequence diagram

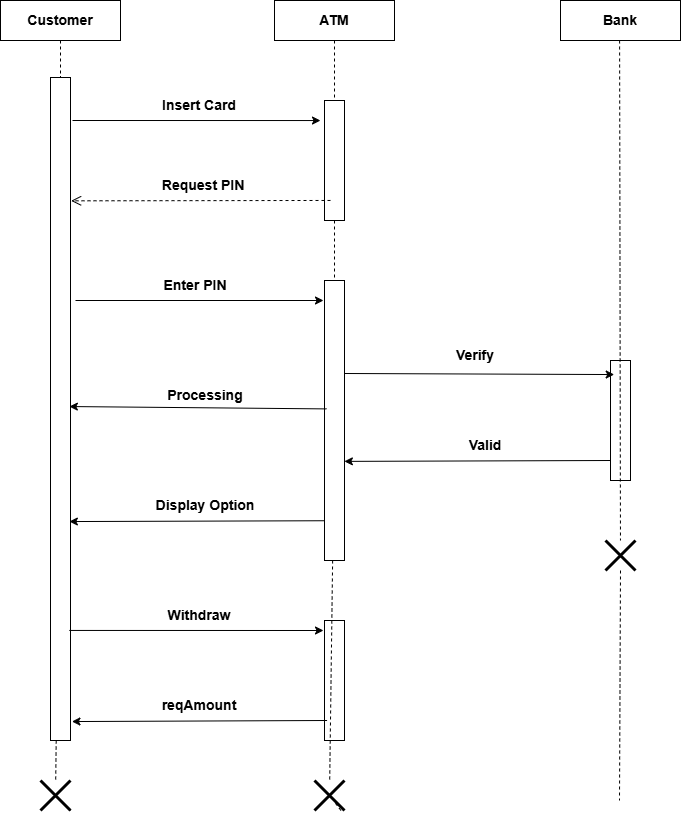


Figure 2.2‑1 Sequence diagram for ATM

A diagram of a customer

Description automatically generated

Figure 2.2‑2 Sequence diagram for Customer Registration

## State diagram

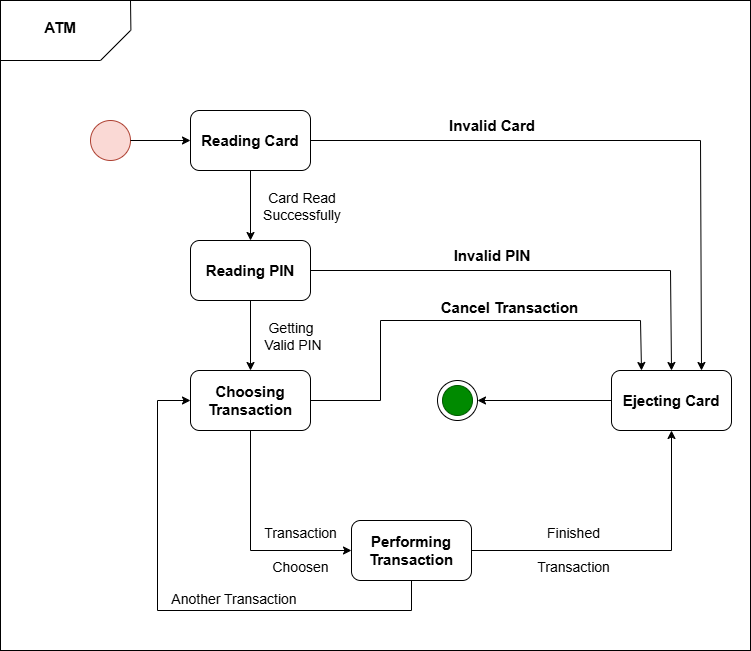


Figure 2.3‑1 State diagram of ATM

## Class diagram

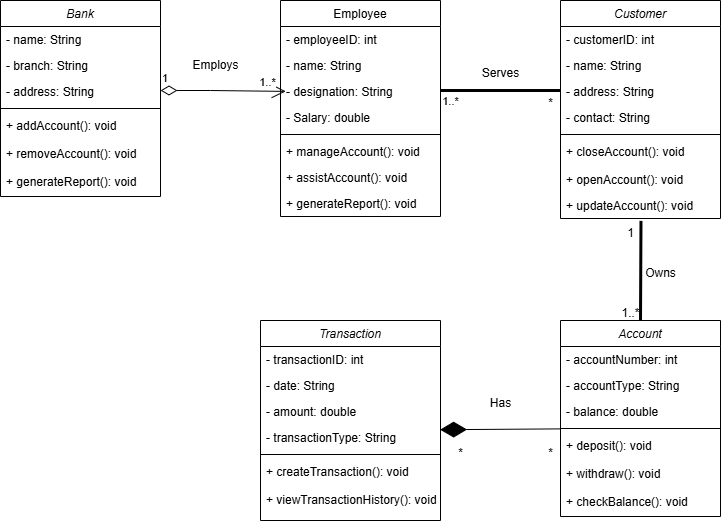


Figure 2.4‑1 Class diagram for Bank management system

## Data flow diagram

### Context diagram (level-0)

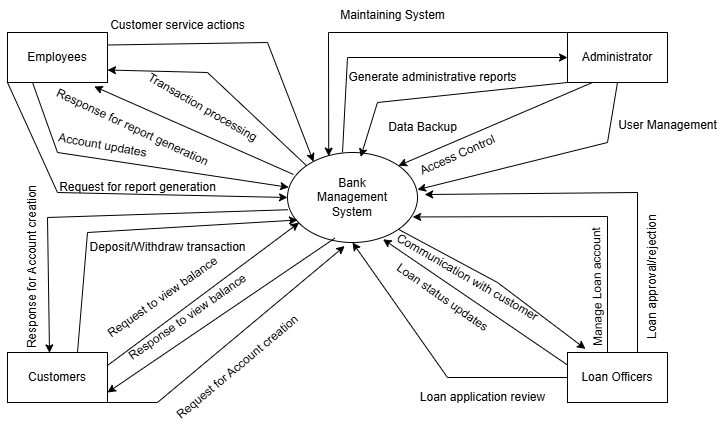


Figure 2.5‑1 Context diagram for Bank management system

### DFD Level-1

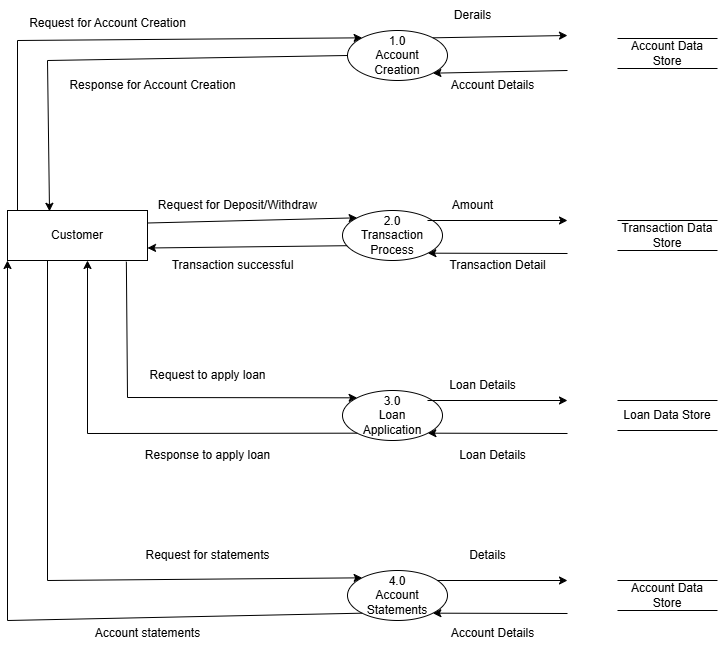


Figure 2.5‑2 DFD level-1 for Bank management system

# External interface requirement (Screens)

## Screen-1: Registration New Account

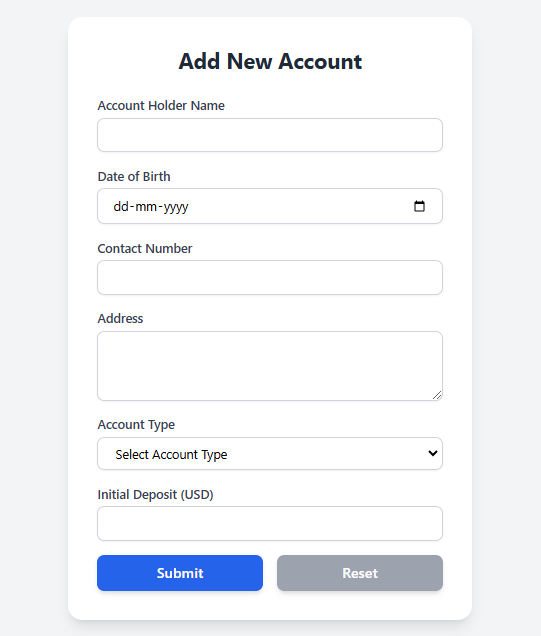


Figure 3.1‑1 Screen-1: Registration New Account

**Purpose:** The purpose of this form is to allow customers to open a new account by providing their personal and account-related details. This ensures the required information is collected to create an account in the system.

Table 3.1‑1 Screen element of Registration New Account

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.** | **Screen Element** | **Input Type** | **O/M** | **1/N** | **Description** |
| 1 | Account Holder Name | Textbox | M | 1 | Editable field to enter the full name of the account holder. |
| 2 | Date of Birth | Date Picker | M | 1 | Field to select the date of birth of the account holder. |
| 3 | Contact Number | Textbox | M | 1 | Editable field to enter the phone number of the account holder. |
| 4 | Address | Text Area | M | 1 | Editable field to enter the address of the account holder. |
| 5 | Account Type | Drop-down | M | 1 | Select the type of account (Savings, Current, or Fixed Deposit) from the list. |
| 6 | Initial Deposit | Number | M | 1 | Editable field to enter the initial deposit amount in USD. |
| 7 | Submit | Button | - | - | Button to store the entered data into the database. |
| 8 | Reset | Button | - | - | Button to clear all fields in the form. |

## Screen-2: Deposit Money Form

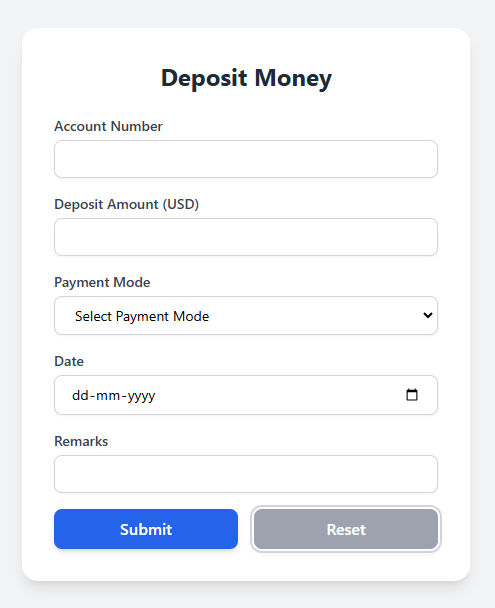


Figure 3.2‑1 Screen-2: Deposit Money Form

**Purpose:** Allow customers to deposit money into their accounts.

Table 3.2‑1 Screen element of Deposit Money Form

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. | Screen Element | Input Type | O/M | 1/N | Description |
| 1 | Account Number | Textbox | M | 1 | Editable field to enter the account number. |
| 2 | Deposit Amount | Number | M | 1 | Editable field to enter the amount to be deposited. |
| 3 | Payment Mode | Drop-down | M | 1 | Select the payment mode (Cash/Cheque/Online) from the list. |
| 4 | Date | Date Picker | M | 1 | Field to pick the date of deposit. |
| 5 | Remarks | Textbox | O | 1 | Optional field to add remarks for the transaction. |
| 6 | Submit | Button | --- | --- | Button to store the entered data into the database. |
| 7 | Reset | Button | --- | --- | Button to clear all fields. |

## Screen-3: Withdraw Money

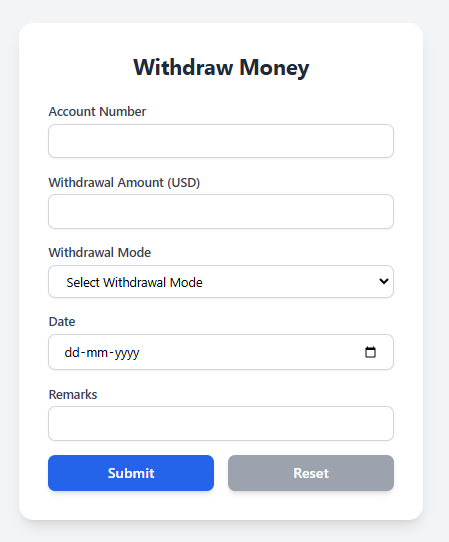


Figure 3.3‑1 Screen-3: Withdraw Money

**Purpose:** Facilitate withdrawal of money from accounts.

Table 3.3‑1 Screen element of Withdraw Money

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. | Screen Element | Input Type | O/M | 1/N | Description |
| 1 | Account Number | Textbox | M | 1 | Editable field to enter the account number. |
| 2 | Withdrawal Amount | Number | M | 1 | Editable field to enter the amount to be withdrawn. |
| 3 | Withdrawal Mode | Drop-down | M | 1 | Select the withdrawal mode (Cash/Online) from the list. |
| 4 | Date | Date Picker | M | 1 | Field to pick the date of withdrawal. |
| 5 | Remarks | Textbox | O | 1 | Optional field to add remarks for the transaction. |
| 6 | Submit | Button | --- | --- | Button to process the withdrawal and store the data into the database. |
| 7 | Reset | Button | --- | --- | Button to clear all fields. |

## Screen-3: Account Statement

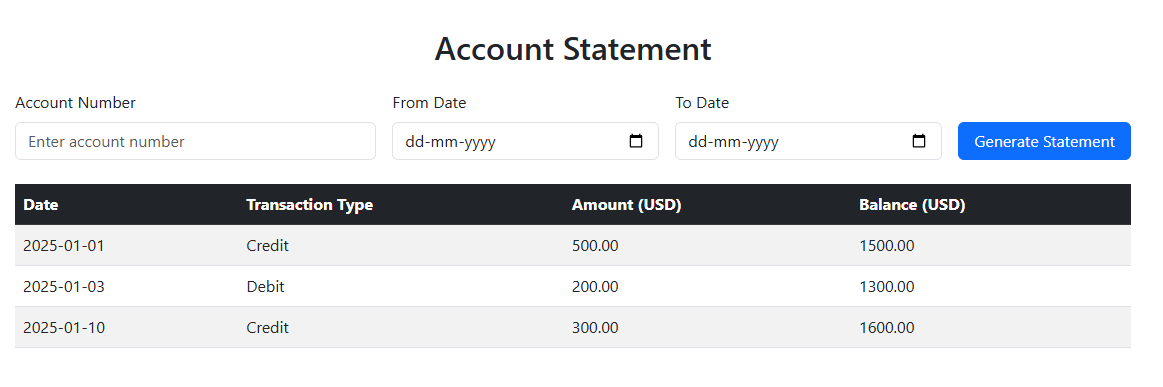


Figure 3.4‑1 Screen-3: Account Statement

**Purpose:** Display account statements based on account number and date range.

Table 3.4‑1 Screen element of Account Statement

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. | Screen Element | Input Type | O/M | 1/N | Description |
| 1 | Account Number | Textbox | M | 1 | Editable field to enter the account number. |
| 2 | Start Date | Date Picker | M | 1 | Field to pick the start date of the statement range. |
| 3 | End Date | Date Picker | M | 1 | Field to pick the end date of the statement range. |
| 4 | Generate Statement | Button | --- | --- | Button to fetch and display the account statement. |

## Screen-3: Fund Transfer



Figure 3.5‑1 Screen-3: Fund Transfer

**Purpose:** Facilitate funds transfer between accounts.

Table 3.5‑1 Screen element of Fund Transfer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. | Screen Element | Input Type | O/M | 1/N | Description |
| 1 | Sender Account Number | Textbox | M | 1 | Editable field to enter the sender's account number. |
| 2 | Receiver Account Number | Textbox | M | 1 | Editable field to enter the receiver's account number. |
| 3 | Amount | Number | M | 1 | Editable field to enter the amount to transfer. |
| 4 | Remarks | Textbox | O | 1 | Optional field to add remarks for the transaction. |
| 5 | Submit | Button | --- | --- | Button to initiate the funds transfer and store the data. |
| 6 | Reset | Button | --- | --- | Button to clear all fields. |

## Screen-3: Loan Services

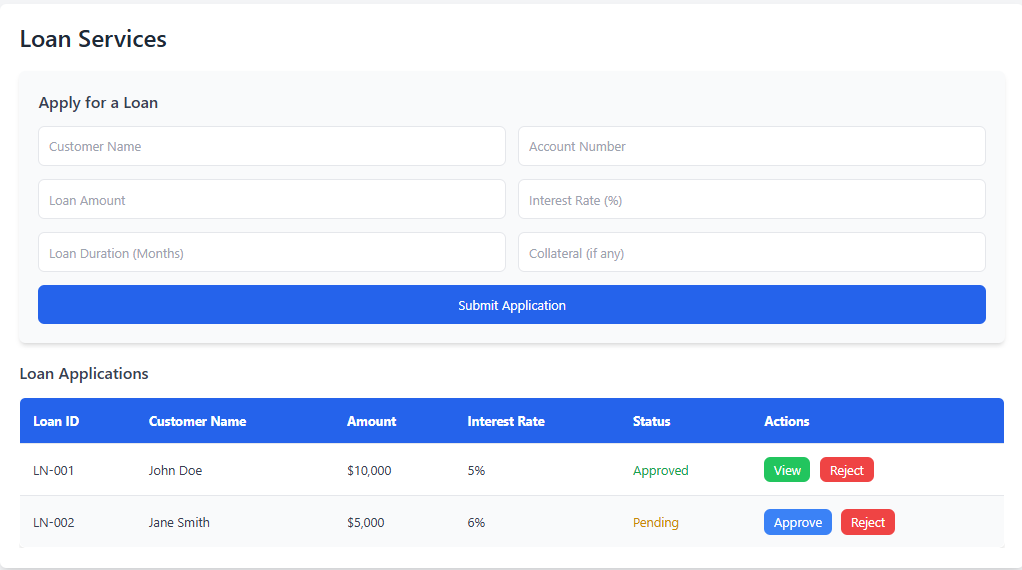


Figure 3.5‑1 Screen-3: Loan Services

**Purpose:** The **Loan Services Screen** facilitates loan applications, management, and approvals within the bank management system. It enables users to submit loan requests, track their status, and manage loan applications efficiently.

Table 3.5‑1 Screen element of Loan Services

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. | Screen Element | Input Type | O/M | 1/N | Description |
| 1 | Customer Name | Textbox | M | 1 | Editable field to enter the applicant's name. |
| 2 | Account Number | Textbox | M | 1 | Editable field to enter the customer’s account number. |
| 3 | Loan Amount | Number | M | 1 | Editable field to enter the requested loan amount. |
| 4 | Interest Rate (%) | Number | M | 1 | Editable field to specify the loan interest rate. |
| 5 | Loan Duration (Months) | Number | M | 1 | Editable field to set the loan duration. |
| 6 | Collateral (if any) | Textbox | O | 1 | Optional field to specify collateral details. |
| 7 | Submit | Button | --- | --- | Button to submit the loan application. |
| 8 | Loan List Table | Table | --- | N | Displays a list of loan applications, including Loan ID, Customer Name, Amount, Interest Rate, Status, and Actions. |
| 9 | Approve Button | Button | --- | 1 | Approves a pending loan application. |
| 10 | Reject Button | Button | --- | 1 | Rejects a loan application. |

## Screen-3: Card Services

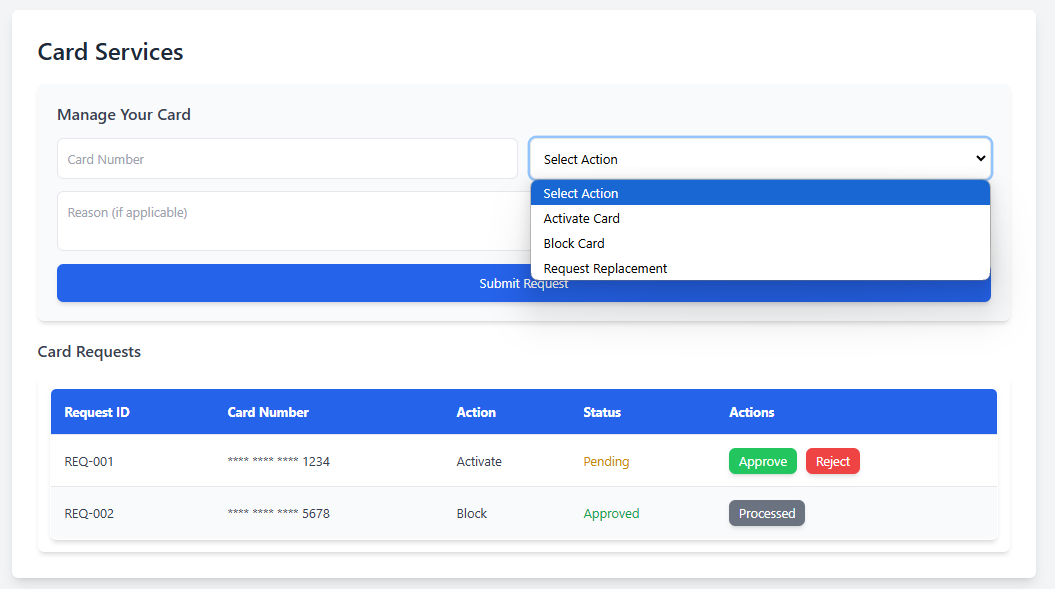


Figure 3.5‑1 Screen-3: Card Services

**Purpose:** The **Card Services Screen** facilitates the management of debit and credit cards, allowing users to activate, block, or request replacements. This ensures secure and efficient card handling for customers.

Table 3.5‑1 Screen element of Card Services

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. | Screen Element | Input Type | O/M | 1/N | Description |
| 1 | Card Number | Textbox | M | 1 | Editable field to enter the card number. |
| 2 | Action Selection | Dropdown | M | 1 | Dropdown to select an action (Activate, Block, Replace). |
| 3 | Reason (Optional) | Textarea | O | 1 | Optional field to provide a reason for the action. |
| 4 | Submit Request | Button | --- | --- | Button to submit the selected card action. |
| 5 | Card Requests Table | Table | --- | N | Displays all card-related requests with status and actions. |
| 6 | Approve Button | Button | --- | 1 | Approves a pending card request. |
| 7 | Reject Button | Button | --- | 1 | Rejects a card request. |

## Screen-3: Fixed Deposit Services

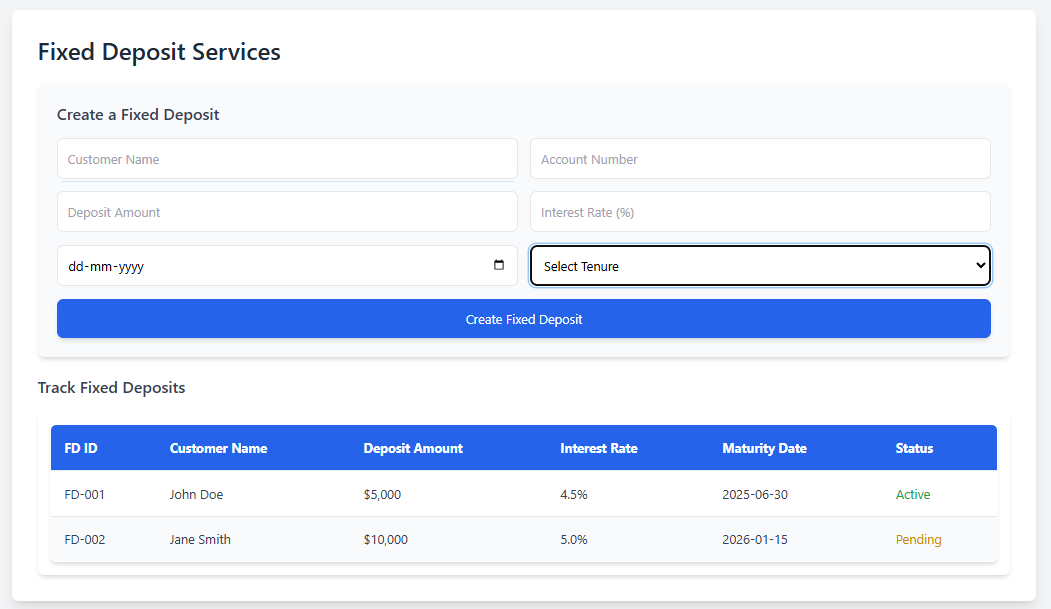


Figure 3.5‑1 Screen-3: Fixed Deposit Services

**Purpose:** The **Fixed Deposit Services Screen** allows bank employee to create and manage fixed deposits. It enables bank employee to enter details for new deposits and track existing ones, ensuring an organized and easy-to-use platform for managing fixed deposit accounts.

Table 3.5‑1 Screen element of Fixed Deposit Services

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. | Screen Element | Input Type | O/M | 1/N | Description |
| 1 | Customer Name | Textbox | M | 1 | Editable field to enter the customer's name. |
| 2 | Account Number | Textbox | M | 1 | Editable field to enter the customer’s account number. |
| 3 | Deposit Amount | Number | M | 1 | Editable field to enter the deposit amount for the fixed deposit. |
| 4 | Interest Rate (%) | Number | M | 1 | Editable field to enter the interest rate for the fixed deposit. |
| 5 | Maturity Date | Date | M | 1 | Editable field to select the maturity date for the fixed deposit. |
| 6 | Tenure Selection | Dropdown | M | 1 | Dropdown to select the tenure of the fixed deposit (6 months, 1 year, 2 years, 5 years). |
| 7 | Create Fixed Deposit Button | Button | --- | --- | Button to submit the form and create the fixed deposit. |
| 8 | Fixed Deposit Table | Table | --- | N | Displays a list of created fixed deposits with their details (ID, Name, Deposit Amount, Interest Rate, Maturity Date, Status). |
| 9 | Status (Active/Pending) | Text | --- | N | Displays the status of each fixed deposit (e.g., Active, Pending). |

# Database design

## List of Tables

* Customers
* Accounts
* Transactions
* Employees
* Loans

Table 4.1‑1 Table: Customers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Data Type | Null | Keys & Constrains | Default Value & Description |
| CustomerID | int | NN | PK (Auto Increment) |  |
| FirstName | varchar(100) | NN |  |  |
| LastName | varchar(100) | NN |  |  |
| Gender | varchar(50) | AN |  |  |
| DateOfBirth | date | NN |  |  |
| ContactNumber | varchar(10) | NN |  |  |
| EmailAddress | varchar(100) | AN |  |  |
| Address | varchar(255) | AN |  |  |

Table 4.1‑2 Table: Accounts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Data Type | Null | Keys & Constrains | Default Value & Description |
| AccountID | int | NN | PK (Auto Increment) |  |
| CustomerID | int | NN | FK | References Customers |
| AccountType | varchar(50) | NN |  |  |
| Balance | decimal(15,2) | NN |  |  |
| CreatedDate | DateTime | NN |  |  |
| Status | varchar(50) | NN |  |  |

Table 4.1‑3 Table: Transactions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Data Type | Null | Keys & Constrains | Default Value & Description |
| TransactionID | int | NN | PK (Auto Increment) |  |
| AccountID | int | NN | FK | References Accounts |
| TransactionType | varchar(50) | NN |  |  |
| Amount | decimal(15,2) | NN |  |  |
| TransactionDate | DateTime | NN |  |  |
| Description | Varchar(255) | AN |  |  |

Table 4.1‑4 Table: Employees

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Data Type | Null | Keys & Constrains | Default Value & Description |
| EmployeeID | int | NN | PK (Auto Increment) |  |
| FirstName | varchar(100) | NN |  |  |
| LastName | varchar(100) | NN |  |  |
| JobTitle | varchar(100) | NN |  |  |
| Department | varchar(100) | NN |  |  |
| EmailAddress | varchar(100) | AN |  |  |
| HireDate | DateTime | NN |  |  |
| Salary | decimal(15,2) | NN |  |  |

Table 4.1‑4 Table: Loans

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Data Type | Null | Keys & Constrains | Default Value & Description |
| LoanID | int | NN | PK (Auto Increment) |  |
| CustomerID | int | NN | FK | References Customers |
| LoanAmount | decimal(15,2) | NN |  |  |
| LoanType | varchar(50) | NN |  |  |
| InterestRate | decimal(5,2) | NN |  |  |
| LoanTerm | int | NN |  |  |
| DisbursedDate | DateTime | NN |  |  |
| Status | varchar(50) | NN |  |  |

# Stories and Scenario

## Story-1: Open a New Bank Account

|  |  |  |
| --- | --- | --- |
| *Story # S1* | : | As a Bank Employee,  I want to Open a new bank account for customer,  So that they can access banking service |
| Priority | **:** | High |
| Estimate | **:** | XL |
| Reason | **:** | Account creation is essential for customers to access financial services. |

### Scenario# S1.1

|  |  |  |
| --- | --- | --- |
| *Scenario# S1.1* | : | Open a New Account with Valid Information |
| Prerequisite | **:** | Bank Employee is logged into the Bank Management System (BMS). |
| Acceptance Criteria | **:** | **Given:** The employee navigates to the account creation page and enters valid details such as name, ID, address, and initial deposit amount.  **When:** The employee submits the form.  **Then:** The system successfully creates the account and displays a confirmation message with the account number. |

### Scenario# S1.2

|  |  |  |
| --- | --- | --- |
| *Scenario# S1.2* | : | Open a New Account with Invalid Information |
| Prerequisite | **:** | Bank Employee is logged into the Bank Management System (BMS). |
| Acceptance Criteria | **:** | **Given:** The employee attempts to create an account with missing or incorrect details (e.g., invalid ID or incomplete address).  **When:** The employee submits the form.  **Then:** The system displays error messages for the incorrect or missing information, and the account is not created. |

## Story-2: Transfer Funds Between Accounts

|  |  |  |
| --- | --- | --- |
| *Story # S2* | : | As a Bank Customer,  I want to transfer funds between my accounts or to another customer’s account,  So that I can easily move money when needed. |
| Priority | **:** | High |
| Estimate | **:** | M |
| Reason | **:** | Fund transfers are essential for smooth financial transactions. |

### Scenario# S2.1

|  |  |  |
| --- | --- | --- |
| *Scenario# S1.1* | : | Transfer Funds with Sufficient Balance |
| Prerequisite | **:** | Customer is logged into the Bank Management System. |
| Acceptance Criteria | **:** | **Given:** The customer navigates to the fund transfer page, selects the source and destination accounts, and enters an amount within the available balance.  **When:** The customer confirms the transfer.  **Then:** The system deducts the amount from the source account, adds it to the destination account, and displays a success message. |

### Scenario# S2.2

|  |  |  |
| --- | --- | --- |
| *Scenario# S1.2* | : | Transfer Funds with Insufficient Balance |
| Prerequisite | **:** | Customer is logged into the Bank Management System. |
| Acceptance Criteria | **:** | **Given:** The customer tries to transfer more funds than available in the source account.  **When:** The customer confirms the transfer.  **Then:** The system displays an error message indicating insufficient funds, and the transfer is not completed. |

## Story-3: Generate Account Statements

|  |  |  |
| --- | --- | --- |
| *Story # S3* | : | As a Bank Customer,  I want to generate my account statements,  So that I can track my transactions. |
| Priority | **:** | Medium |
| Estimate | **:** | M |
| Reason | **:** | Customers need to review their transaction history for financial management. |

### Scenario# S3.1

|  |  |  |
| --- | --- | --- |
| *Scenario# S1.1* | : | Generate a Statement for a Specific Period |
| Prerequisite | **:** | Customer is logged into the Bank Management System. |
| Acceptance Criteria | **:** | **Given:** The customer navigates to the account statement page and selects a specific date range.  **When:** The customer requests the statement.  **Then:** The system generates and displays the statement with all transactions within the selected period. |

### Scenario# S3.2

|  |  |  |
| --- | --- | --- |
| *Scenario# S1.2* | : | Attempt to Generate a Statement for an Invalid Period |
| Prerequisite | **:** | Customer is logged into the Bank Management System. |
| Acceptance Criteria | **:** | **Given:** The customer selects an invalid date range (e.g., a future date).  **When:** The customer requests the statement.  **Then:** The system displays an error message indicating an invalid date range, and no statement is generated. |

# Test cases

|  |  |  |  |
| --- | --- | --- | --- |
| Project Name: | EMI Calculator | Test Designed by: | P. U. Jadeja |
| Module Name: | **Login** | **Test Designed date:** | 01-10-2023 |
| Release Version: | **1.0** | **Test Executed by:** | **R. B. Gondaliya** |
|  |  | **Test Execution date:** | 15-01-2023 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pre-condition: Web application should be accessible | | | | |
| Test Case ID | **Test Title** | **Test Type** | **Description** | **Test Case ID** |
| TC\_001 | Login to web application with valid credential | Functional | Login to Library management system web application through valid credential | TC\_001 |
| TC\_002 | Login to web application with invalid credential | Functional | Login to Library management system web application through invalid credential | TC\_002 |
| TC\_003 | Varify login page elements | GUI | varify that all elements are availabe on login page | TC\_003 |

|  |  |
| --- | --- |
| **Test Case Title** | Login to web application with valid credential |
| **Test Type** | Functional |
| **Test Priority** | High |
| **Pre-condition** | Web application should be accessible |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Step** | **Test Case Description** | **Expected Result** | **Actual Result** | **Status** | **Comment** | **Data** | **BUG ID** |
| 1 | Access Web application URL | The site launched properly | Site launched successfully | Pass |  | <https://accounts.google.com/ServiceLogin> |  |
| 2 | Enter valid Username in username field | Username field should be editable and accept the Username | Username input accepted | Pass |  | Username:  Rbgondaliya@gmail.com |  |
| 3 | Enter valid Password in Password field | Password field should be editable and accept the password and display as star or dot | Password input displayed in dot and accepted | pass |  | Password: rbgondaliya |  |
| 4 | Enter valid captcha code in captch field | Captch field should editable and accept captcha and captcha is case sensitive | Captcha input accepted | Pass | Step required when human action validation perform | get captcha from image which is near by captcha field |  |
| 5 | Click on login button | User should login into site and navigated to dashboard | User navigated to dashboard and username should br display in top of the right side. | pass |  |  |  |

|  |  |
| --- | --- |
| **Test Case Title** | Login to web application with invalid credential |
| **Test Type** | Functional |
| **Test Priority** | Medium |
| **Pre-condition** | Web application should be accessible |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Step** | **Test Case Description** | **Expected Result** | **Actual Result** | **Status** | **Comment** | **Data** | **Bug ID** |
| 1 | Verify that User is not able to Login with invalid Username and invalid Password | Should be display an error message enter wrong username or password | Display an error of wrong username and password | Pass |  |  |  |
| 2 | Verify that User is not able to Login with Valid Username and invalid Password | Should be display an error message enter wrong password | Display an error of wrong password | Pass |  |  |  |
| 3 | Verify that User is not able to Login with invalid Username and Valid Password | Should be display an error message User not found | Display an error Username not found | Pass |  |  |  |
| 4 | Verify that User is not able to Login with blank Username or Password | Set required field validation message for Username and Password | Display an error of wrong username and password | Fail | Not performa a validation function fix it |  | Bug\_002 |

|  |  |
| --- | --- |
| **Test Case Title** | Varify login page elements |
| **Test Type** | GUI |
| **Test Priority** | Medium |
| **Pre-condition** | Web application should be accessible |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Step** | **Test Case Description** | **Expected Result** | **Actual Result** | **Status** | **Comment** | **Data** | **Bug ID** |
| 1 | Launch application with the given url | The site launched properly | Site launched successfully | Pass |  | <https://accounts.google.com/ServiceLogin> |  |
| 2 | Verify that the login screen contains elements such as Username, Password, Sign in button, Remember password check box, Forgot password link, and Create an account link. | All listed control displayed properly on the page | Login page loaded successfully | Pass |  |  |  |
| 3 | Verify that cursor is focused on “Username” text box on the page load | Cursor is focused in Username textbox | Cursor focus in Username textbox | Pass |  |  |  |
| 4 | Verify that tab functionality is working properly or not | When tab pressed cursor move in next control | Cursor moving in next control | Pass |  |  |  |
| 5 | Verify that all the fields such as Username, Password has a valid placeholder | All text fields have proper placeholder | All text fields have proper placeholder | Pass |  |  |  |
| 6 | Verify that the labels float upward when the text field is in focus or filled (In case of floating label) | When field is focused or filled, label display on top of the filled | When field is focus or filled, label display on top of the filled | Pass | step required when fields with floating label |  |  |
| 7 | verify that forgot password link working properly | when click on forgot password load forgot passworg page | forgot password link not working | Fail |  |  |  |

# References

* http://www.w3schools.com/html/html\_intro.asp
* https://www.w3schools.com/php/default.asp
* https://www.javatpoint.com/uml