Title: Online Bank Management System

- **Subtitle:** Ex. 1 Analysis and Identification of Suitable Process Models using Bizagi Modeler
- **Name:** Karan Sehgal
- **Registration No:** 22BCE3939
- **Team No:**
- **Course/Subject:** Software Engineering Lab (BCSE301P)
- **Instructor’s Name:** Dr. Mehfooza M
- **Date of Submission:** 01/01/25

Introduction:

The Online Bank Management System is designed to centralize banking operations, optimize customer experiences, and ensure secure and efficient financial transactions. This system serves three key stakeholders: **customers**, **tellers**, and **administrators**, each with specific access levels and functionalities. The goal is to provide a user-friendly platform that aligns with modern technological standards and adheres to industry best practices.

System Requirements:

I have divided System Requirements into two sub categories:

Functional Requirements

These describe the service that the banking management system must offer, they are subdivided into three access levels: Admin Mode, Teller Mode, and Customer Mode:

Customer Mode

This mode allows customers to perform various self-service operations related to their accounts.

1. **Sign in with login and password:**
    - Customers authenticate themselves using a unique **username** and **password**.
    - Implement **Multi-Factor Authentication (MFA)** for enhanced security (e.g., SMS OTP, email verification).
2. **Update personal details:**
    - Modify profile information such as:
        - Name
        - Address
        - Contact information (phone, email)
    - Changes should require validation through a secure channel.
3. **Change password:**
    - Customers can reset their passwords through:
        - Old password verification.
        - Email or SMS-based confirmation for security.
        - Enforced strong password policies.
4. **View balance:**
    - Real-time display of account balances.

    - Separate views for **savings**, **current**, or other types of accounts.

5. **View personal history of transactions:**
    - Display a list of recent transactions, including:
        - Date and time.
        - Transaction type (debit/credit).
        - Amount and remaining balance.
        - Transaction remarks or references.
    - Enable filtering and searching (e.g., by date range or type).
6. **Transfer money:**
    - Internal and external transfers:
        - **Internal Transfers:** Between accounts in the same bank.
        - **External Transfers:** To accounts in other banks using NEFT, RTGS, or IMPS.
    - Include safeguards:
        - Daily transfer limits.
        - Beneficiary management with approval.
7. **Withdraw:**
    - Request withdrawals online for in-branch collection or linked ATM cards.
    - Option to generate a withdrawal slip or virtual token.
8. **Submit cash:**
    - Log cash deposits for in-branch processing or through automated cash deposit machines.
    - Generate receipts for records.

Manager Mode

Managers facilitate customer banking operations and manage accounts with specific permissions.

1. **Sign in with login and password:**
    - manager-specific credentials with role-based access controls.
    - Monitor logins and actions for audit purposes.
2. **Change password:**
    - Ability to reset their own passwords, adhering to strong password policies.
3. **Register new bank customers:**
    - Create new customer profiles by capturing:

        - Personal information (e.g., name, address, contact details).
        - Government-issued ID verification (e.g., KYC compliance).
        - Account preferences (e.g., savings, current).
    - Assign initial account credentials.
4. **View customer information:**
    - Retrieve customer profiles, including:
        - Personal and contact details.
        - Account types and balances.
        - Transaction history.
5. **Manage customer accounts:**
    - Perform actions such as:
        - Activate or deactivate accounts.
        - Update account details (e.g., account type changes).
        - Process requests for checkbooks, debit/credit cards.
        - Handle customer issues like account freezes.

Admin Mode

Admins manage the system’s overall structure, including user roles, branches, and operational settings.

1. **Sign in with login and password:**
    - Admins authenticate with high-security credentials.
    - Two-factor authentication (TFA) is mandatory.
2. **View manager and customer details:**
    - Access detailed views of:
        - Teller accounts (e.g., roles, assigned branches, activity logs).
        - Customer accounts (e.g., balances, transaction histories).
        - Generate reports for managerial review.
3. **Add or update bank branch details:**
    - Manage branch information:
        - Create new branch records with location, contact details, and codes.
        - Modify existing branch data as needed.
4. **Add or update manager details:**
    - Manage teller accounts:
        - Create or modify teller profiles.
        - Assign branches or roles.

- Set permissions and access levels

Common Functionalities Across Modes

1. **Audit Logging:**
   - Record all activities performed by users for accountability.
   - Include timestamps, user IDs, and action descriptions.
2. **Session Management:**
   - Implement secure session handling (e.g., session timeout, token invalidation).
3. **Notifications:**
   - Real-time notifications for actions like password changes, balance updates, or account modifications.
4. **Dashboard View:**
   - Personalized dashboards showing relevant KPIs(Key performance indicator), alerts, or pending actions for each user type.

Non Functional Requirements

The **non-functional requirements** for an **Online Bank Management System (OBMS)** ensure that the system performs efficiently, reliably, and securely, meeting the broader quality attributes critical for banking applications. These requirements are just as vital as functional requirements and focus on system performance, usability, security, and maintainability.

1. Performance Requirements

- **Latency:**
  - System should process transactions (e.g., balance inquiry, fund transfer) in less than 2 seconds.
- **Scalability:**
  - Handle at least 10,000 concurrent users during peak load with minimal performance degradation.
- **Throughput:**
  - Process at least 500 transactions per second (TPS) for real-time operations

2. Security Requirements

- **Data Security:**
  - Encrypt all sensitive data (e.g., account details, passwords) using AES-256 encryption.
- **Authentication:**
  - Implement multi-factor authentication (MFA) for customer and admin logins.
- **Fraud Detection:**
  - Real-time anomaly detection to identify and block suspicious activities.
- **Compliance:**
  - Ensure compliance with industry standards like PCI-DSS and GDPR.

3. Availability and Reliability

- **Uptime:**
  - Guarantee 99.99% system availability to minimize downtime.
- **Disaster Recovery:**
  - Implement a disaster recovery system to restore operations within 15 minutes of failure.
- **Fault Tolerance:**
  - Ensure the system can handle hardware or software failures without impacting critical functions.

4. Usability Requirements

- **User Experience:**
    - Ensure an intuitive interface for customers, tellers, and admins.
    - Provide user assistance (e.g., FAQs, chatbots) for self-service operations.
- **Accessibility:**
    - Ensure compliance with WCAG 2.1 standards for accessibility to support users with disabilities.

5. Maintainability and Modularity

- **Code Maintainability:**
    - Ensure code is modular and adheres to coding standards for easy updates.
- **Version Upgrades:**
    - Allow seamless integration of new features or system updates with minimal disruption.
- **Error Logging and Monitoring:**
    - Include centralized logging and monitoring to identify and resolve issues quickly.

6. Auditability

- **Transaction Logs:**
    - Maintain detailed logs for all transactions for at least 7 years.
- **Access Logs:**
    - Record all login attempts and actions performed by users for audit and compliance.
- **Audit Trail:**
    - Provide a tamper-proof audit trail for all system activities.

7. Legal and Regulatory Compliance

- **Regulations:**
    - Adhere to Anti-Money Laundering (AML), Know Your Customer (KYC), and other applicable banking regulations.
- **Data Privacy:**
    - Ensure customer data is stored and processed in compliance with local laws (e.g., GDPR, CCPA).

2.Justification of Process Model:

I feel the software development lifecycle model (SDLC) which best suits my project the most is AGILE. You may ask **why does Agile work for Banking systems?**

When creating a banking system, it's not just about writing code to make a software—it's about building something secure, flexible, and user-friendly. With all the moving parts in a bank system—customers, tellers, admins, regulations—things can change quickly. Agile is perfect for this because it's all about adapting, collaborating, and delivering value step by step. The literal sense of the word agile means flexible. It forms the basis of one of the most fast and responsive to change SDLCs which puts customer over how things are being done by taking constant feedback. It prioritizes close customer collaboration over contract negotiation, involving them throughout the development process to make sure that the final product aligns with their ever-changing needs. Contracts define boundaries but real value comes from ongoing dialogue and partnership.

Lets break it down to better understand Why Agile?

1. Adapting to Changes

Imagine you're halfway through building the system, and suddenly, new banking regulations require biometric login or a new government protocol. If you're following a rigid plan (i.e a iterative model or worse a waterfall model), this is a nightmare. But with Agile, you're working in short cycles (sprints), so you can adapt quickly. You prioritize the new feature, work on it in the next sprint, and keep going without throwing everything else off balance.

For Our Online Banking Management System, this means:

- Regulations? No problem, we adjust as needed.
- New customer demands? They're added to the next sprint.

2. Delivering Early and Often

Instead of waiting months (or years) to finish the entire system, Agile gets you started with the basics first. Picture this: in the first few sprints, you build login functionality and a simple dashboard where customers can check their balance. It's not the full system yet, but it's enough to start testing and getting feedback. If the user wants to launch the system as early as possible, it can be done with agile. Later, more advanced features like transaction histories or fraud detection are added incrementally. This way, the system evolves in real-time based on what’s needed most urgently.

This gives user the freedom to access core features early so that they can start benefiting from the system sooner rather than later. This creates space for the development team to welcome some early feedback for improvements ,their input helps shape future iterations, ensuring the system meets their actual needs.

3. Putting the Customer First

What’s the point of building a banking system if it doesn’t work the way people expect? Agile brings customers (and other stakeholders) into the process. After every sprint, you show them what’s been built:

- Customers might say, “The balance view is great, but can you make it easier to find recent transactions?”
- Tellers might add, “We need a faster way to search for a customer’s account.”

You will have to accommodate these grievances in the next sprint.

4. Teamwork Makes the Dream Work or should I say Work a Dream

Building a Bank Management System isn’t a one-person job. Developers, testers, banking experts, and even compliance officers need to work together. Agile thrives on this collaboration. Everyone sits down during sprint planning to decide what’s most important. If something isn’t working, they regroup in retrospectives to figure out how to improve. At the end of each day

For example:

- A developer might say, “We’re spending too much time fixing login bugs.”
- A tester could respond, “Let’s add more automated tests to catch these earlier.”

This openness and teamwork ensure the system is high-quality and everyone’s on the same page.

5. Reducing Risk

Banking systems handle sensitive data, so security and reliability are non-negotiable. Agile helps manage these risks by building and testing in small, manageable pieces.

For instance:

- After adding the login feature, it's immediately tested for security vulnerabilities.
- As you introduce transaction processing, you stress-test it to ensure it works under heavy loads.

By catching issues early, you avoid big, scary failures down the line.

Lets take a **case study** to better understand the justification:

Wells Fargo's Agile Journey

- **Background**: Wells Fargo embraced Agile to improve digital banking services and enhance customer experience.
- **Implementation**:
  - Introduced Agile processes in its product development lifecycle, including regular sprints and stakeholder reviews.
  - Focused on iterative improvements to its mobile app and online platforms.
- **Results**:
  - Frequent updates to the mobile app, adding features like personalized alerts and streamlined account management.
  - Enhanced fraud detection systems with real-time notifications and faster response mechanisms.

Wells Fargo's success with Agile isn't just a case study—it's proof that banking systems thrive in an Agile environment. By embracing short iterations, customer feedback, and collaboration, Wells Fargo transformed its processes and delivered better results.

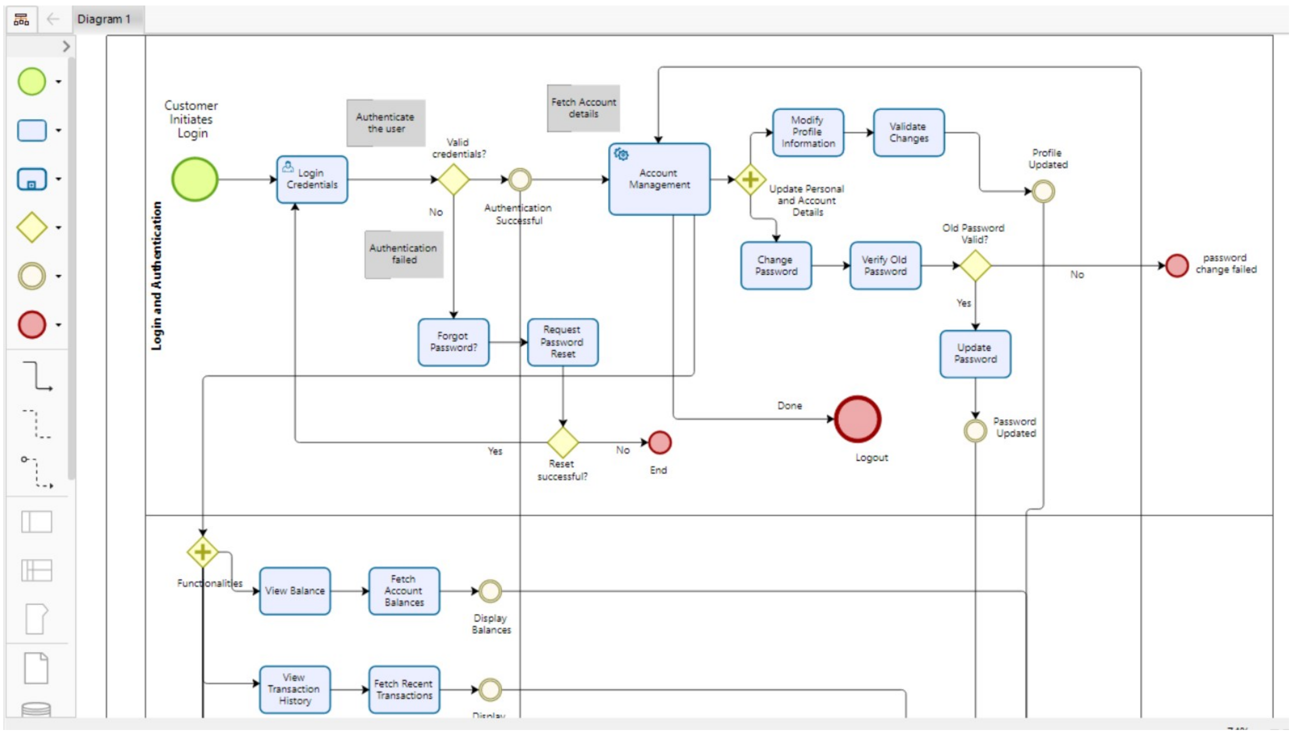
For my project, Agile ensures:

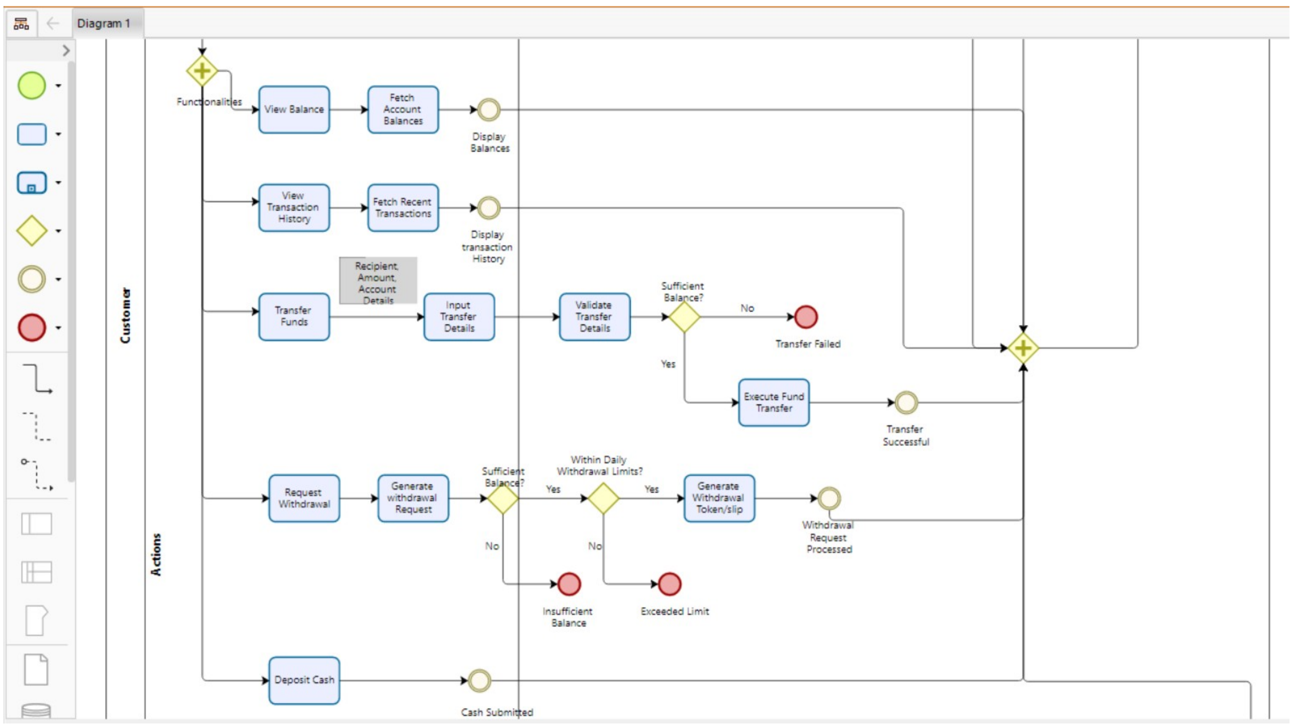
- Rapid adaptation to industry changes.
- Incremental delivery of features to build trust and value.
- A system that evolves with its users, just like Wells Fargo's innovations.

The lesson from Wells Fargo is clear: Agile doesn't just build software—it builds better banking systems.

3. Process Diagram Output:

Customer View





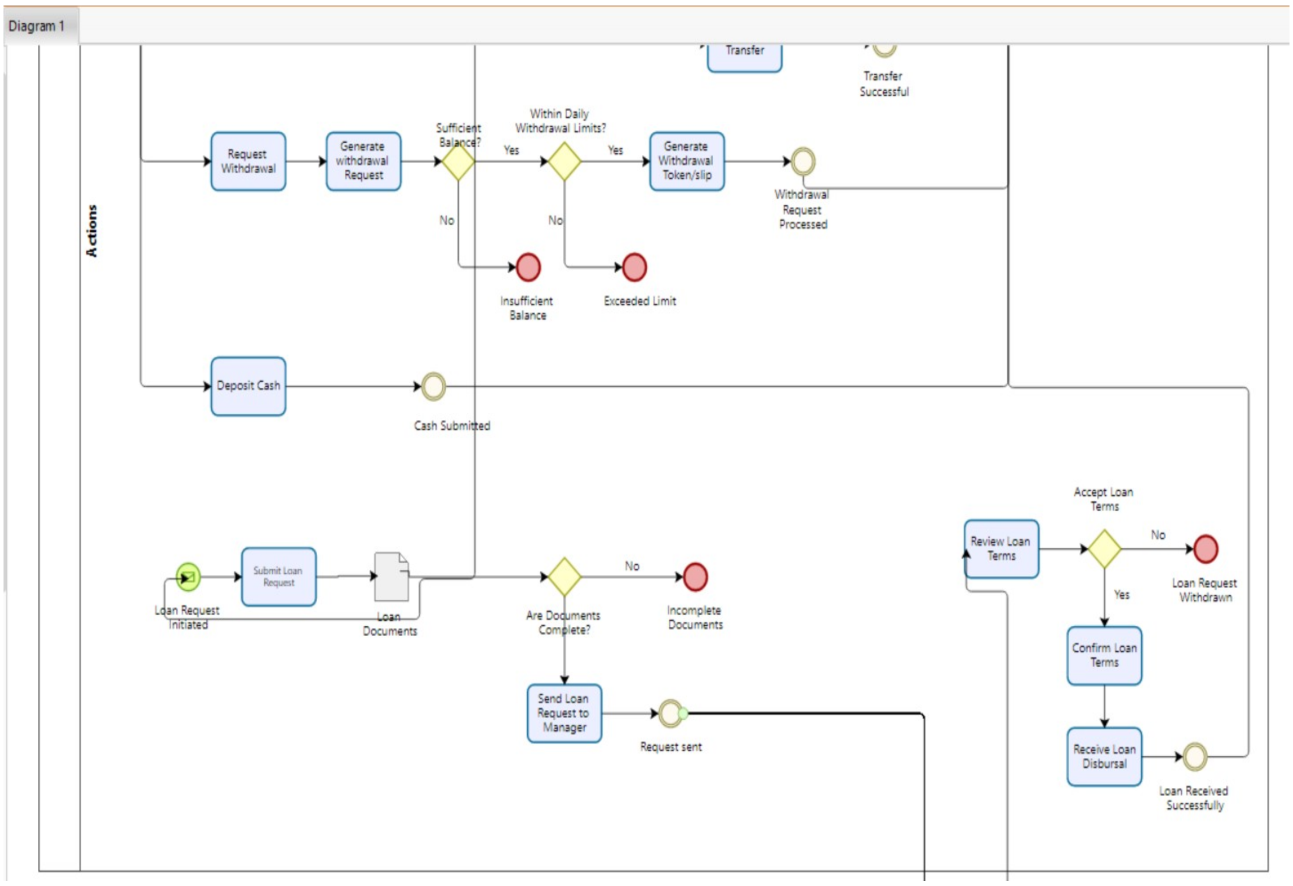
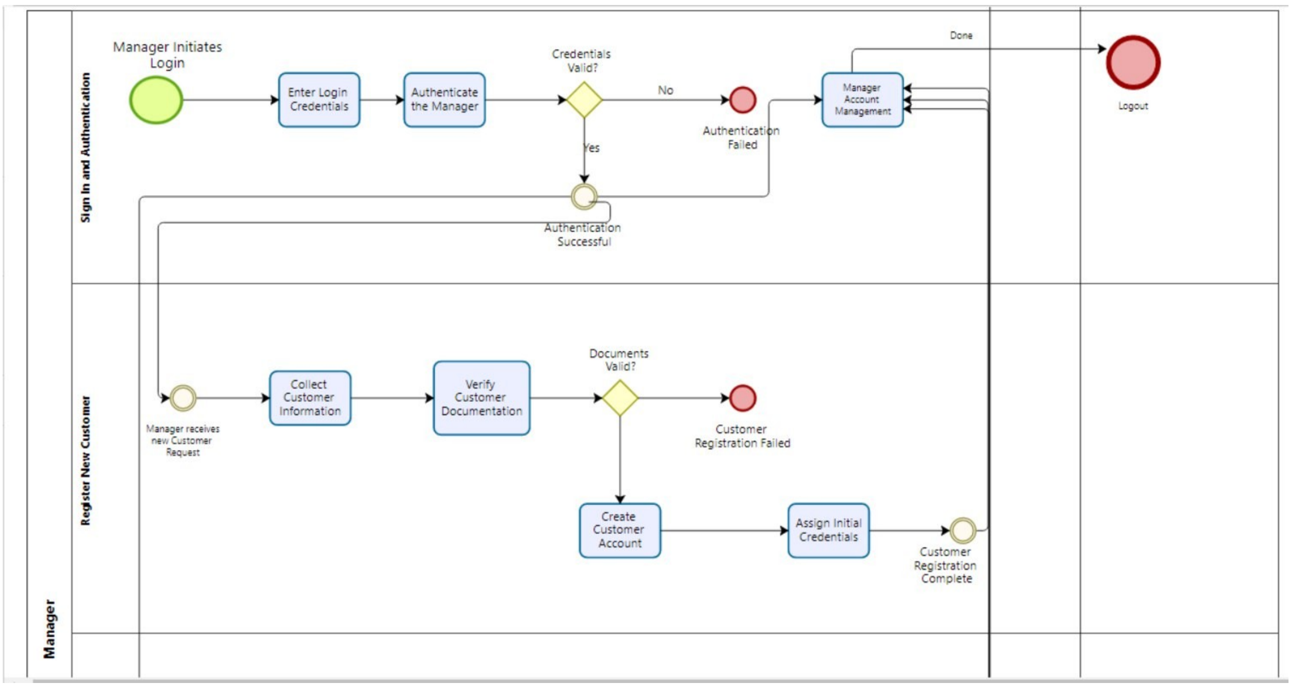
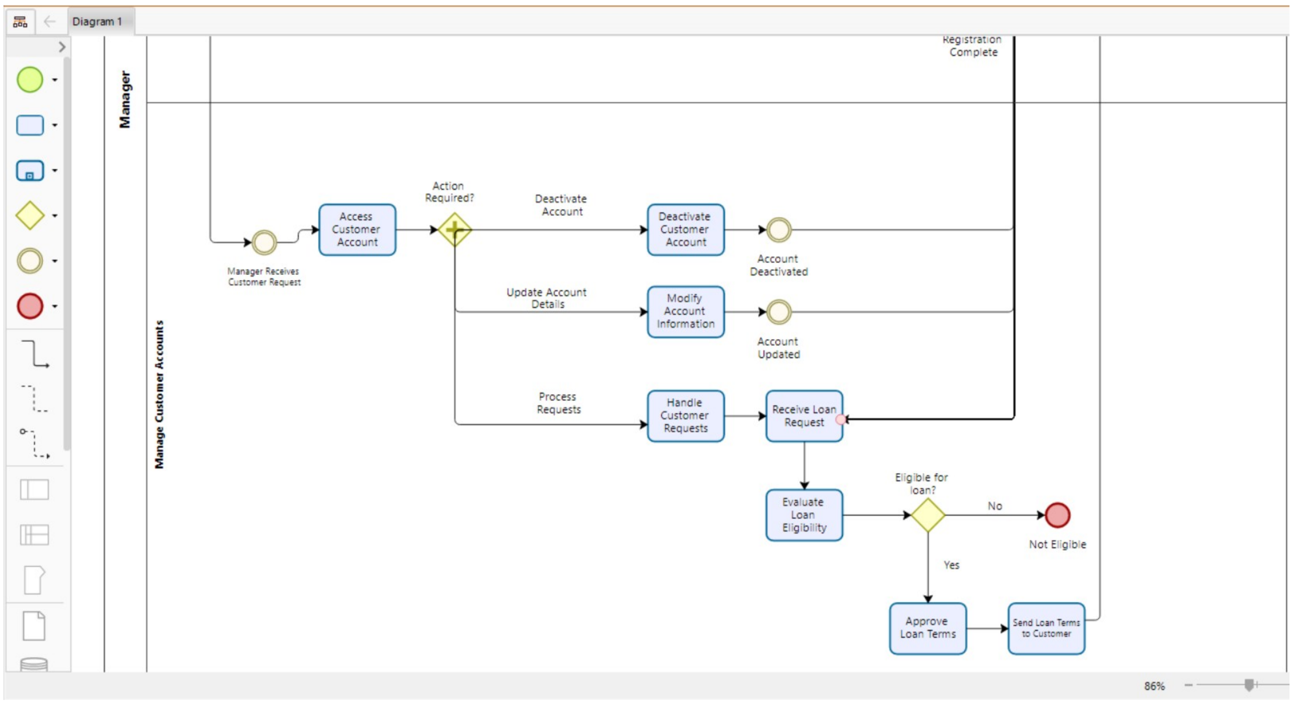


Diagram 1
Actions
Transfer
Transfer Successful
Request Withdrawal
Generate withdrawal Request
Sufficient Balance?
Yes
Within Daily Withdrawal Limits?
Yes
Generate Withdrawal Token/slip
Withdrawal Request Processed
No
No
Insufficient Balance
Exceeded Limit
Deposit Cash
Cash Submitted
Loan Request Initiated
Submit Loan Request
Loan Documents
Are Documents Complete?
No
Incomplete Documents
Send Loan Request to Manager
Request sent
Review Loan Terms
Accept Loan Terms
No
Loan Request Withdrawn
Yes
Confirm Loan Terms
Receive Loan Disbursal
Loan Received Successfully

Manager View


Manager
Sign In and Authentication
Register New Customer
Manager Initiates Login
Enter Login Credentials
Authenticate the Manager
Credentials Valid?
No
Yes
Authentication Failed
Authentication Successful
Manager Account Management
Done
Logout
Manager receives new Customer Request
Collect Customer Information
Verify Customer Documentation
Documents Valid?
Customer Registration Failed
Create Customer Account
Assign Initial Credentials
Customer Registration Complete



Admin View

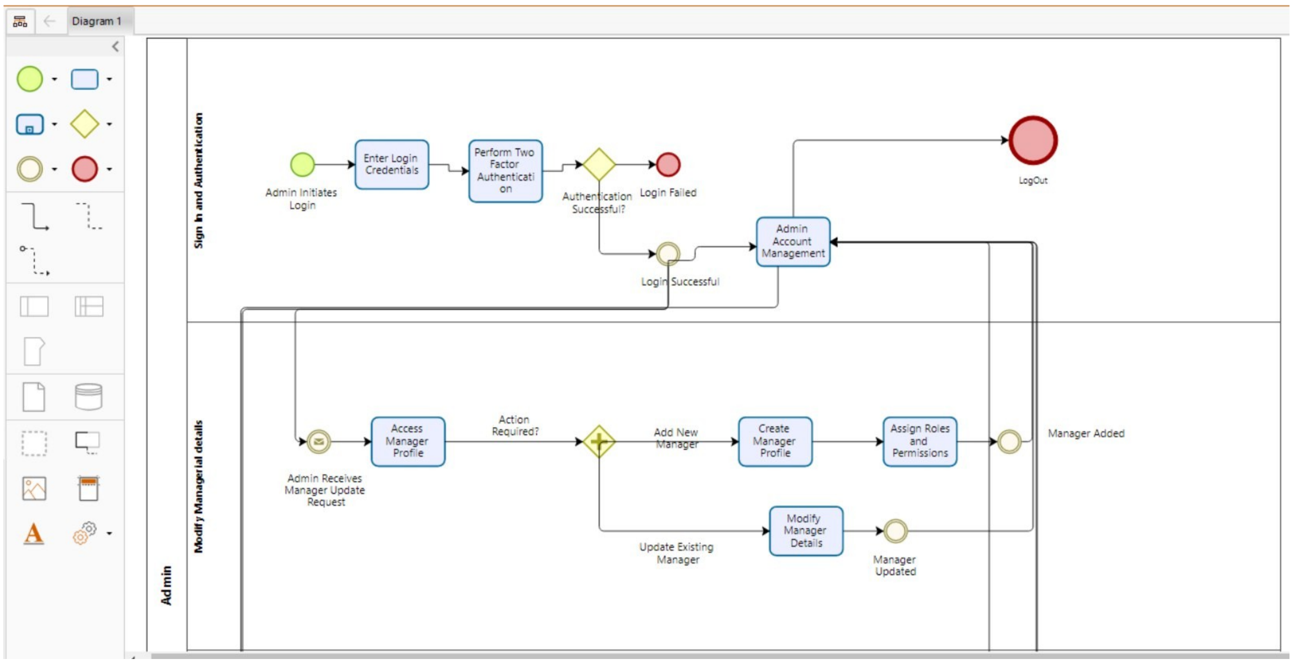


Diagram 1
Admin
Sign in and Authentication
Admin Initiates Login
Enter Login Credentials
Perform Two Factor Authentication
Authentication Successful?
Login Failed
Login Successful
Admin Account Management
LogOut
Modify Managerial details
Admin Receives Manager Update Request
Access Manager Profile
Action Required?
Add New Manager
Create Manager Profile
Assign Roles and Permissions
Manager Added
Update Existing Manager
Modify Manager Details
Manager Updated

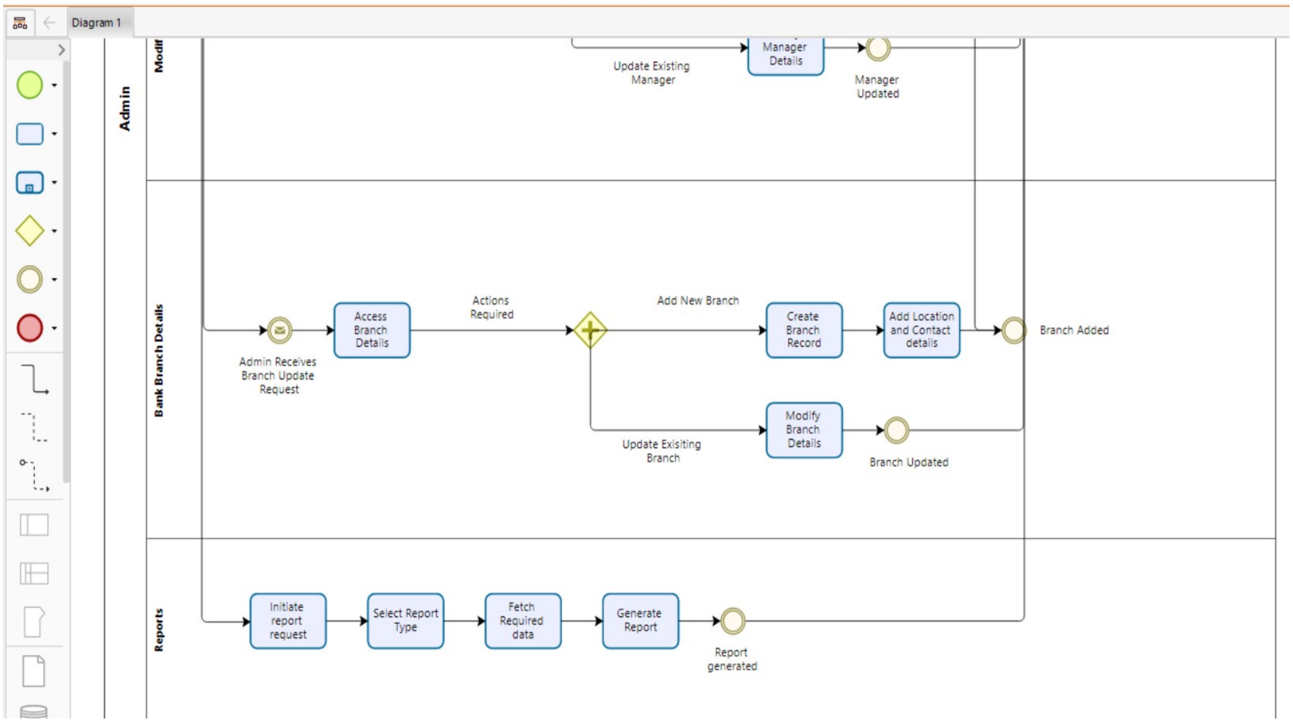


Diagram 1
Admin
Modif
Update Existing Manager
Manager Details
Manager Updated
Bank Branch Details
Admin Receives Branch Update Request
Access Branch Details
Actions Required
Add New Branch
Create Branch Record
Add Location and Contact details
Branch Added
Update Exisiting Branch
Modify Branch Details
Branch Updated
Reports
Initiate report request
Select Report Type
Fetch Required data
Generate Report
Report generated

4. Key Flow

Here’s the **key flow** for each stakeholder:

Customer Flow

1. **Login/Authentication:**
    - Input username and password.
    - Optional: Multi-Factor Authentication (MFA).
2. **Perform Actions:**
    - View balance and transaction history.
    - Transfer funds to another account.
    - Update personal details or reset password.
3. **Logout:**
    - Securely terminate the session.

Manager Flow

1. **Login/Authentication:**
    - Input teller credentials.
    - Securely verify access levels.
2. **Perform Actions:**
    - Register a new customer.
    - View and update customer account details.
    - Handle deposits, withdrawals, or account closures.
3. **Logout:**
    - Securely log out to prevent unauthorized access.

Admin Flow

1. **Login/Authentication:**
    - Admin-specific credentials with high-security protocols.
2. **Perform Actions:**
    - Add or modify bank branch details.
    - View and manage teller and customer accounts.
    - Generate system reports or perform audits.
3. **Logout:**
    - Ensure complete session termination.