BANKING MANAGEMENT SYSTEM

1. Introduction

1.1 Purpose

The purpose of this SRS document is to outline the requirements for the Banking Management System (BMS), which simulates core functionalities of a bank such as customer account management, transactions, loan processing, and administrative oversight. The system provides secure login mechanisms and role-based access to ensure data consistency, security, and the prevention of race conditions through file management and synchronization.

1.2 Intended Audience

- **Software Developers:** For understanding the technical requirements to implement the system.
- **Banking Personnel:** Bank employees, managers, and administrators who will interact with the system.
- **System Administrators:** For setting up and maintaining the system.

1.3 Intended Use

The system will be used to manage banking operations, including:

- Customer account management.
- Transaction handling and loan processing.
- Role-based access for different types of users (Customer, Employee, Manager, Administrator).

1.4 Product Scope

The Banking Management System is designed to:

- Handle multiple users concurrently through secure login mechanisms.
- Provide role-based functionalities for customers, employees, managers, and administrators.
- Ensure data consistency and security through file locking, multithreading, and synchronization mechanisms.

2. Overall Description

2.1 Product Perspective

The BMS uses a client-server architecture where the server manages databases and multiple clients access the system. The server handles user requests and provides appropriate access based on user roles.

2.2 Assumptions and Dependencies

- The system is developed using the C programming language.
- Clients and servers are connected over a reliable network.
- The server hosting the BMS has sufficient resources to manage multiple concurrent clients.

3. System Architecture

The architecture of the Banking Management System is designed to facilitate seamless interactions between clients and the server, ensuring efficient processing of banking operations while maintaining data integrity and security. The system follows a client-server model, where the client interfaces with users and the server manages data and operations.

3.1 Client Component

The client component is responsible for user interaction. Customers, bank employees, managers, and administrators access the system through a graphical user interface (GUI) or command-line interface (CLI), depending on their roles. Each user logs in with unique credentials, establishing a secure session. The client sends requests to the server for various operations, such as viewing account balances or processing loan applications. The client also handles user input, displays data received from the server, and ensures that all transactions are conducted securely.

3.2 Server Component

The server component acts as the backbone of the Banking Management System. It maintains the database and processes client requests concurrently. The server utilizes socket programming to establish communication channels with multiple clients, allowing it to handle simultaneous connections efficiently. Each client-server interaction is managed through dedicated sockets, ensuring that data is transmitted securely and reliably.

3.3 Socket Programming

Socket programming is a crucial aspect of the Banking Management System's architecture. It enables bidirectional communication between the client and server, allowing clients to send requests and receive responses seamlessly. The server listens for incoming connections on a specific port, and when a client connects, a new socket is created for that session. This approach allows the server to manage multiple client sessions without interference, ensuring that each user's operations are processed independently.

4 Functional Requirements

4.1 Customer Functionalities

1. **Login System:** Allow customers to log in securely with their credentials. Only one active session per user is permitted.
2. **View Account Balance:** Customers can view their account balance.
3. **Deposit Money:** Enable customers to deposit funds into their accounts.
4. **Withdraw Money:** Allow customers to withdraw funds from their accounts.
5. **Transfer Funds:** Facilitate fund transfers between customer accounts.
6. **Apply for a Loan:** Provide a loan application feature for customers.
7. **Change Password:** Allow customers to change their login passwords.
8. **Add Feedback:** Enable customers to submit feedback.
9. **View Transaction History:** Display a customer's transaction history.

10. **Logout:** Securely log out of the system.
11. **Exit:** End the session.

4.2 Employee Functionalities

1. **Login System:** Secure login with one session per user.
2. **Add New Customer:** Employees can register new customers.
3. **Modify Customer Details:** Update existing customer information.
4. **Process Loan Applications:** Handle loan application processing.
5. **Approve/Reject Loans:** Decide on loan applications.
6. **View Assigned Loan Applications:** Access specific loan applications assigned to the employee.
7. **View Customer Transactions:** Display customer transactions in a passbook-style interface.
8. **Change Password:** Allow employees to update their passwords.
9. **Logout:** Securely log out.
10. **Exit:** Terminate the session.

4.3 Manager Functionalities

1. **Login System:** One session per user.
2. **Activate/Deactivate Customer Accounts:** Manage account statuses.
3. **Assign Loan Application Processes to Employees:** Distribute loan applications for processing.
4. **Review Customer Feedback:** Monitor feedback submitted by customers.
5. **Change Password:** Allow password updates for managers.
6. **Logout:** Securely log out.
7. **Exit:** End the session.

4.4 Administrator Functionalities

1. **Login System:** One session per user.
2. **Add New Bank Employee:** Administrators can register new employees.
3. **Modify Customer/Employee Details:** Update details for both customers and employees.
4. **Manage User Roles:** Assign or modify roles for users.
5. **Change Password:** Allow administrators to update their passwords.
6. **Logout:** Securely log out.
7. **Exit:** End the session.

5. Non-Functional Requirements

5.1 Performance Requirements

- The system should support up to 100 concurrent users.
- Response times for operations should be under 2 seconds.

5.2 Security Requirements

- Passwords should be securely encrypted and stored.
- File locking mechanisms should prevent data inconsistencies due to concurrent access.
- Role-based access must be strictly enforced.

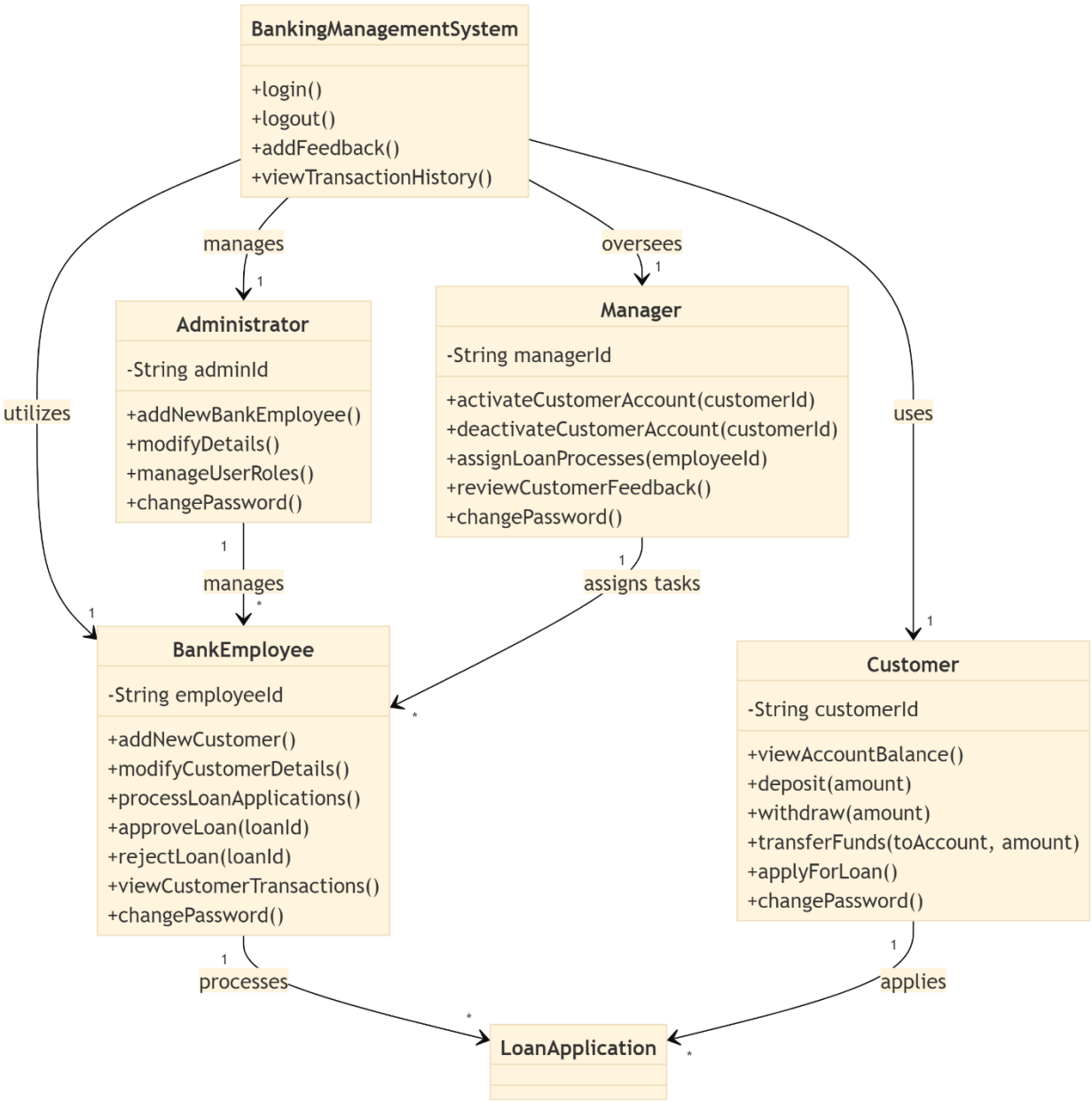
5.3 Reliability Requirements

- The system should maintain data integrity with proper synchronization mechanisms.
- Backup and recovery features should be available to handle server failures.

5.4 Usability Requirements

- The system should have a user-friendly interface, with clear menus and options based on user roles.

6. System Design:



7. Functional Design

Each user role is encapsulated within a class, allowing for a clear definition of responsibilities and interactions. The `User` class serves as a base class, from which specialized roles—**`Customer`**, **`BankEmployee`**, **`Manager`**, and `Administrator`—inherit common attributes and methods like **`login`** and **`logout`**.

Customer Class: Customers can perform banking operations such as viewing account balances, making deposits, and applying for loans. The methods ensure that transactions are processed atomically, mitigating race conditions through effective locking mechanisms.

BankEmployee Class: This class enables employees to manage customer accounts and loan processes. Features such as **`addCustomer`** and **`processLoan`** allow employees to efficiently handle their operational tasks while maintaining secure access to customer data.

Manager Class: Managers oversee the operations of employees, with functionalities that include activating and deactivating accounts. They can also assign loan applications to employees, ensuring a smooth workflow and responsiveness to customer needs.

Administrator Class: Administrators possess the highest level of access, managing user roles and ensuring compliance across the system. Their functionalities allow for the addition of new employees and the modification of user details, reinforcing the system's security and operational integrity.

8. Glossary

- **Client-Server Architecture:** A model where clients request services from a central server.

- **Race Condition:** A situation where multiple processes try to access shared resources simultaneously, potentially leading to data inconsistencies.
- **ACID Properties:** Principles that ensure reliable processing of transactions (Atomicity, Consistency, Isolation, Durability).