**Software Requirements Specification (SRS)**

**Project Title: Console-Based Banking Application**

**Version:** 1.0  
**Date:** 7-11-2024

**Authors:** Sri Karthika L

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to outline the requirements for a console-based banking application designed to simulate basic banking operations. This system will allow users to create and manage accounts, perform transactions, view account details, and generate reports. The application will be implemented using Java with a relational database for persistent storage.

**1.2 Scope**

This application will support core banking functionalities such as account management, transaction processing, and report generation. Additionally, it will utilize Java concepts such as Object-Oriented Programming, Collections, Exception Handling, File Handling, JDBC for database interaction, and multithreading to ensure thread-safe operations.

**1.3 Definitions, Acronyms, and Abbreviations**

* **JDBC:** Java Database Connectivity
* **DBMS:** Database Management System
* **CRUD:** Create, Read, Update, Delete
* **SRS:** Software Requirements Specification

**1.4 References**

* Java Documentation
* MySQL Documentation
* JDBC API Documentation

**2. Overall Description**

**2.1 Product Perspective**

The console-based banking application will operate independently and will be executed via the command line. It will interact with a MySQL database for data storage and retrieval.

**2.2 Product Functions**

The banking application will have the following main functionalities:

1. **User Account Management:** Create accounts, view details, and update information.
2. **Transaction Management:** Deposit, withdraw, and transfer funds.
3. **Transaction History Management:** Save and retrieve transaction history from files.
4. **Multithreading and Concurrency:** Thread-safe transactions.
5. **Database Operations (JDBC):** Store and retrieve account and transaction data in a MySQL database.
6. **Reporting:** Generate and display various reports such as customer details, transaction history, and daily summaries.

**2.3 User Characteristics**

The users of this application are primarily bank employees or administrative personnel responsible for managing customer accounts and performing transactions.

**2.4 Constraints**

* The application must use a relational database (e.g., MySQL) for persistent storage.
* Transactions must be thread-safe to avoid data inconsistency.
* Custom exceptions must handle errors such as insufficient balance or invalid account number.
* All reports should be generated using SQL queries to ensure optimal database performance.

**2.5 Assumptions and Dependencies**

* Users have basic knowledge of command-line interfaces.
* The application assumes a secure, single-user console environment for simplicity.
* Java Runtime Environment and MySQL server are pre-installed on the system.

**3. System Requirements**

**3.1 Functional Requirements**

**3.1.1 User Account Management**

* **FR1:** The system shall allow creating a new bank account (Savings, Current).
* **FR2:** The system shall allow users to view account details.
* **FR3:** The system shall allow updating account information such as address and contact number.

**3.1.2 Transaction Management**

* **FR4:** The system shall allow users to deposit money into their accounts.
* **FR5:** The system shall allow users to withdraw money from their accounts.
* **FR6:** The system shall enable users to transfer money between accounts.

**3.1.3 Transaction History**

* **FR7:** The system shall save transaction history to a file for record-keeping.
* **FR8:** The system shall retrieve and display transaction history from a file.

**3.1.4 Multithreading and Concurrency**

* **FR9:** The system shall use multithreading to handle concurrent transactions.
* **FR10:** The system shall ensure thread safety using synchronization to avoid race conditions.

**3.1.5 Exception Handling**

* **FR11:** The system shall handle exceptions for scenarios such as insufficient balance and invalid account numbers.

**3.1.6 Database Operations (JDBC)**

* **FR12:** The system shall store account and transaction information in a database using JDBC.
* **FR13:** The system shall perform CRUD operations on bank accounts.

**3.1.7 Report Generation**

* **FR14:** The system shall generate a report of customer details.
* **FR15:** The system shall provide a transaction history report for a particular account.
* **FR16:** The system shall show the total balance of all accounts in the bank.
* **FR17:** The system shall display the number of accounts created, categorized by Savings and Current accounts.
* **FR18:** The system shall generate a daily summary of deposits, withdrawals, and transfers.

**3.2 Non-Functional Requirements**

**3.2.1 Performance Requirements**

* The system shall execute typical transactions (deposit, withdraw, transfer) within 2 seconds.

**3.2.2 Reliability Requirements**

* The system shall ensure data consistency even under concurrent access.
* The application shall recover gracefully from database connection failures.

**3.2.3 Usability Requirements**

* The system should provide meaningful error messages for invalid operations.

**3.2.4 Security Requirements**

* User credentials and sensitive data should not be stored directly in the application code.

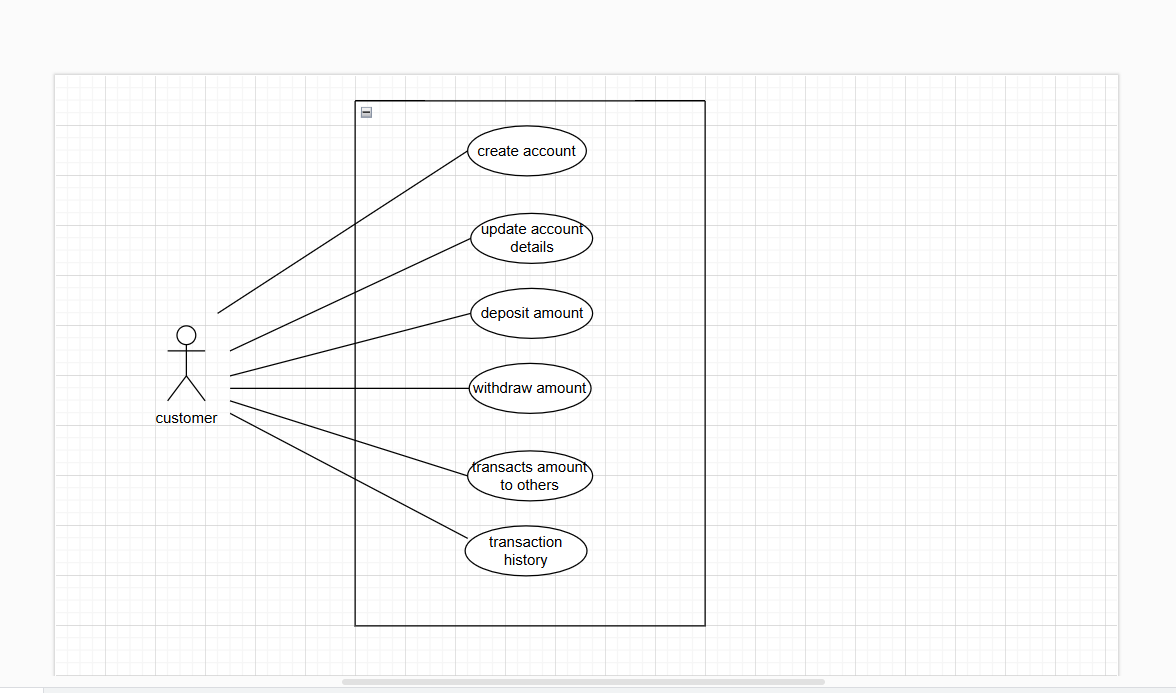
**3.2.5 Maintainability Requirements**

* The system shall be modular and maintainable, with well-documented classes and methods.

**4. System Architecture**

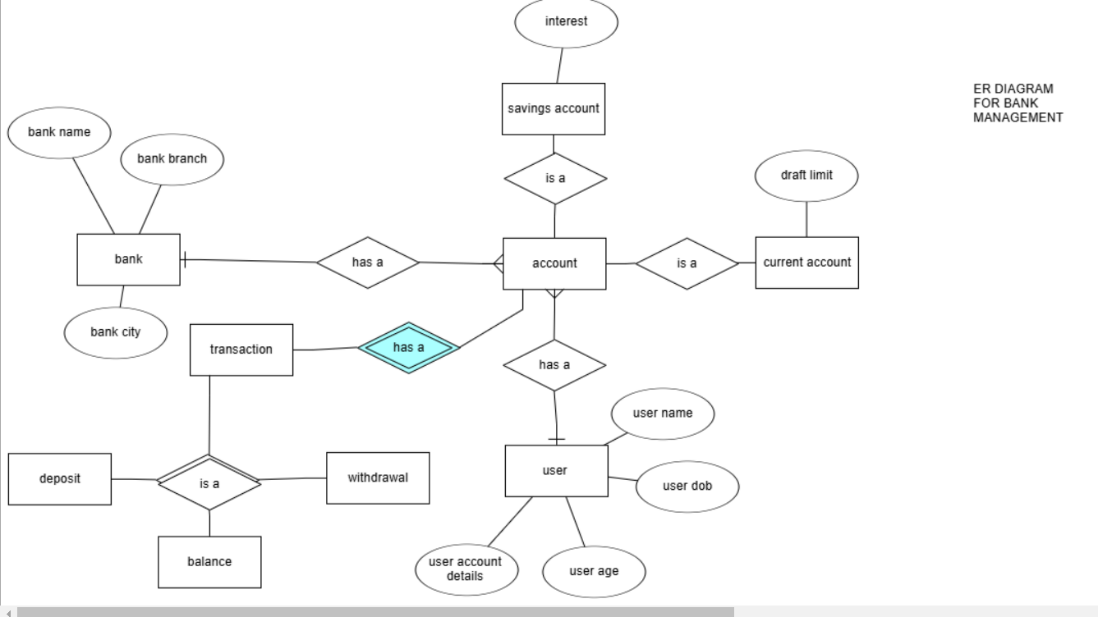
**4.1 Use Case Diagram**

The Use Case diagram here is to visually represent the interactions between users and the system's core functionalities. This will help illustrate the relationships between actors (e.g., bank employee, system) and actions (e.g., creating accounts, making transactions).



**4.2 ER Diagram**

The ER Diagram after the Use Case diagram to provide a detailed representation of the database structure, including entities, attributes, and relationships. The ER diagram is crucial to visualize how data is organized and connected across tables, especially for a project involving database interactions like this one. In this ER diagram the user are the customers(end users).



**4.3 Class Diagram**

The class diagram for this application will include classes such as:

* **Bank**: Manages multiple accounts.
* **Account** (abstract class): Base class for SavingsAccount and CurrentAccount.
* **Transaction** (abstract class): Base class for DepositTransaction, WithdrawalTransaction, and other specific transaction types.

**4.4 Database Schema**

The database schema includes the following tables:

1. **Account Table:** Stores basic account information.
2. **SavingsAccount Table:** Inherits from Account and includes interest\_rate.
3. **CurrentAccount Table:** Inherits from Account and includes overdraft\_limit.
4. **Transaction Table:** Stores transaction records.
5. **DepositTransaction Table:** Stores specific details for deposit transactions.
6. **WithdrawalTransaction Table:** Stores specific details for withdrawal transactions.
7. **Bank Table:** Stores bank information.

**5. External Interface Requirements**

**5.1 User Interfaces**

* Console-based user interface for executing commands and displaying results.

**5.2 Hardware Interfaces**

* Standard input/output devices, no additional hardware required.

**5.3 Software Interfaces**

* **Database**: MySQL for persistent data storage.
* **JDBC**: To facilitate database connectivity.

**6. Other Requirements**

**6.1 Database Requirements**

* The database should ensure ACID properties for transaction management.
* Implement stored procedures for complex tasks such as fund transfers.
* Define triggers to automatically log transactions when balance changes occur.

**6.2 Backup and Recovery (Optional)**

* Regular backups should be implemented to safeguard data integrity.

**7. Appendix**

**7.1 Sample SQL Queries**

* Example SQL queries for creating tables, inserting data, and retrieving reports.

**7.2 Sample Console Outputs**

* Sample outputs for each operation, including account creation, transactions, and report generation.

This SRS document provides a comprehensive foundation for developing the Console-Based Banking Application. You can use it as a guide to implement the project according to the specified requirements.