Project Title	:	**Bank Management Application**
Author	:	Karthikeyan M
Date of Documentation	:	07-11-2024
Technologies	:	Java , DBMS

SRS Document Bank Application

1. Introduction

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Project Scope

1.5 References

2. Overall Description

2.1 Product Perspective

2.2 Product Features

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 Assumptions and Dependencies

3. System Features

3.1 Functional Requirements

3.2 Usecase Diagram

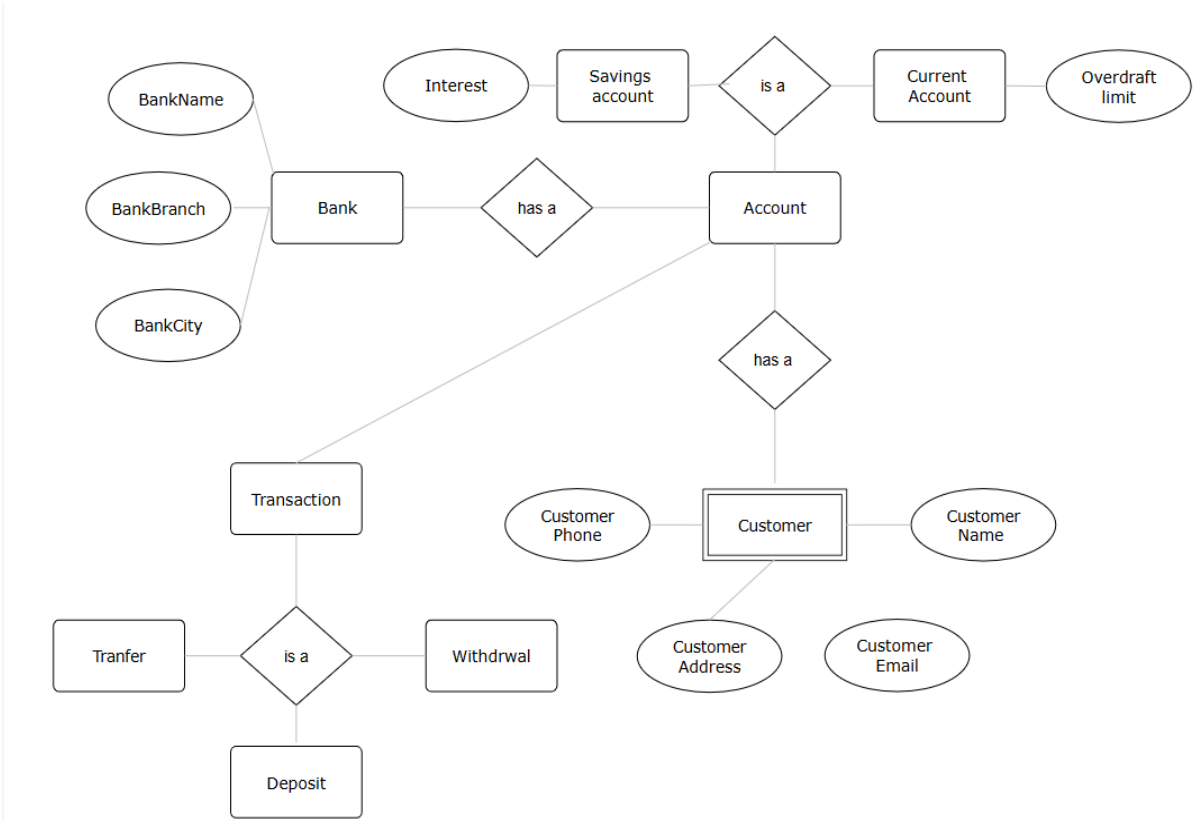
1. Introduction

1.1 Purpose

The purpose of this document is to outline the requirements for a bank application that facilitates various banking functions. This application aims to streamline banking operations for both users and bank employees, providing secure and efficient ways to manage accounts, conduct transactions, and maintain financial records. The document specifies the functionality for managing general banking entities like accounts, including specific types such as savings and current accounts, as well as the operations related to transactions.

1.2 Document Convention

Entity-Relationship Diagram (ERD): Used to visually represent the relationships among key entities (e.g., Bank, Account, Transaction). All ERD elements follow standard notation (rectangles for entities, diamonds for relationships, and ovals for attributes).



1.3 Intended Audience and Reading Suggestions

Intended Audience

- Developers: To understand requirements for building the application.
- Project Managers: To manage project scope and timelines.
- QA Testers: To create and run test cases based on requirements.
- Stakeholders: To ensure business goals are met.

Reading Suggestions

- Developers: Focus on Section 3 (Functional Requirements).
- Project Managers: Start with Sections 1 and 2 for project overview.
- Testers: Refer to Sections 3 and 4 for validation.
- Stakeholders: Read Sections 1 and 2 for a high-level understanding.

1.4 Project Scope

This project aims to develop a bank application that supports the following features:

- **Account Management**: Creation, modification, and maintenance of accounts (savings, current, and other account types).
- **Transaction Management**: Handling deposits, withdrawals, transfers, and transaction histories.
- **User Management**: Secure user authentication and role-based access control (customers, bank employees, and administrators).
- **Reporting and Analytics**: Generation of reports on account balances, transaction history, and other financial metrics.
- **Security**: Implementation of secure login, data encryption, and compliance with regulatory standards (e.g., GDPR, PCI DSS).

The application will be designed to provide a secure, user-friendly platform for managing various banking operations efficiently, ensuring that customers and employees can easily interact with the system.

1.5 References

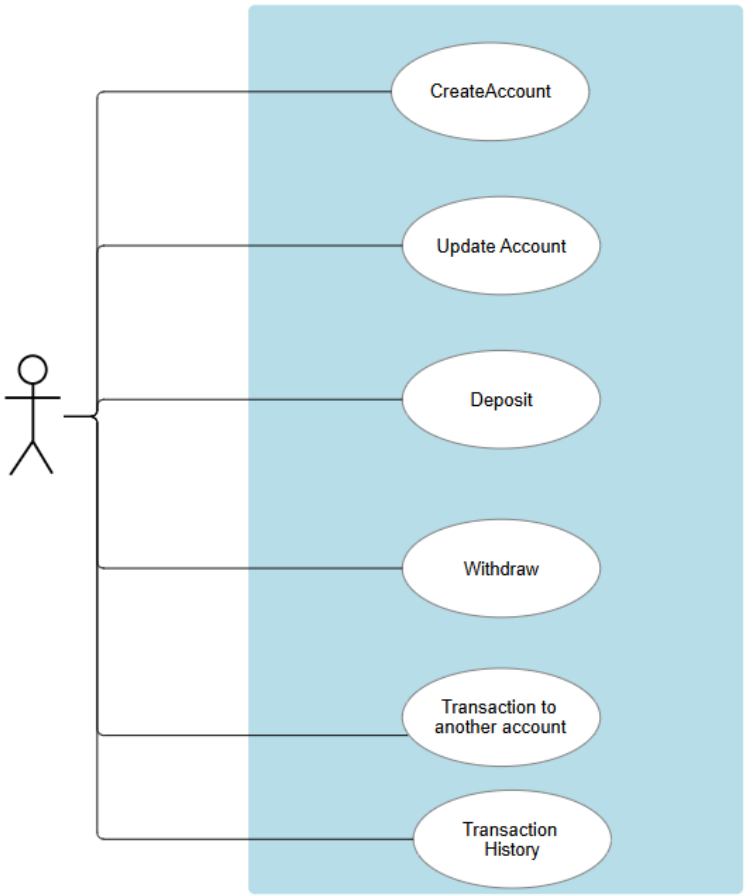
- **SQL Documentation**: Guidelines and best practices for working with SQL databases.

- **JDBC Documentation**: Java Database Connectivity (JDBC) API documentation for connecting Java applications to databases.
- **Java Documentation**: Official documentation for Java programming language and libraries.
- **SmartDraw**: Software documentation for creating diagrams like ERDs (Entity-Relationship Diagrams) and other flowcharts to represent database structure and application processes.

These references will support your development process using SQL, JDBC, Java, and SmartDraw for diagrams.

3.2 Use Case Diagram

The **Use Case Diagram** provides an overview of the major functionalities the system will support. It shows how the users (actors) interact with the system to achieve specific goals. In this case, the system is a simple bank application.



2. Overall Description

2.1 Product Perspective

This is a simple bank application that helps manage customer accounts, process transactions, and generate basic reports. It will be used by customers, bank employees, and administrators.

2.2 Product Features

- Account Management: Create and manage bank accounts (e.g., savings, current).
- Transaction Management: Handle deposits, withdrawals, and transfers.
- User Management: Secure login for customers and bank staff.
- Reporting: Generate basic reports like transaction history.

2.3 User Classes and Characteristics

- Customers: Use the system to manage their accounts and perform transactions.
- Bank Employees**:** Help customers with account issues and generate reports.
- Administrators**:** Manage users and oversee the system.

2.4 Operating Environment

The application will run on a server and be accessible through a web browser. It will use **Java** for backend logic, **MySQL** for storing data, and **JDBC** for database connections.

2.5 Design and Implementation Constraints

- The system will only use MySQL and JDBC**.**
- The design will be simple and user-friendly.
- The application will follow basic security measures to protect user data.

2.6 Assumptions and Dependencies

- Users need internet access to use the application.
- The system will depend on **MySQL** for data storage.
- Future updates may include more features or improvements.

3. System Features

3.1 Functional Requirements

1. **Account Management**

    - FR-1.1**:** The system must allow customers to create new accounts (savings, current).
    - FR-1.2**:** The system must allow customers to view their account details (balance, transaction history).
    - FR-1.3**:** The system must allow bank employees to update customer account details.
    - FR-1.4**:** The system must allow administrators to deactivate or delete accounts.

2. **Transaction Management**

    - FR-2.1**:** The system must allow customers to deposit money into their accounts.
    - FR-2.2**:** The system must allow customers to withdraw money from their accounts.
    - FR-2.3**:** The system must allow customers to transfer money between their own accounts (e.g., savings to current).
    - FR-2.4**:** The system must keep a record of each transaction (date, amount, type of transaction).
    - FR-2.5**:** The system must notify customers by email or SMS after each transaction.

4. External Interface Requirements

4.1 User Interfaces

- **Login Screen**: Customers and employees will enter their username and password to access the system.
- **Account Dashboard**: Displays account details, transaction history, and options to make deposits, withdrawals, or transfers.
- **Transaction** : Allows users to initiate transactions (deposit, withdraw, or transfer funds).

4.2 Hardware Interfaces

- The system will run on standard desktop and mobile devices. No specific hardware is required, but the application needs:
  - A **server** to host the application.
  - **User devices** (PCs, tablets, smartphones) to access the system through a web browser.

4.3 Software Interfaces

- **Database**: The system will connect to a **MySQL** database to store account information, transactions, and user data.
- **Web Browser**: The application will be accessed through a web browser (Chrome, Firefox, Edge).

4.4 Communications Interfaces

- **Internet Connection**: The system will require an internet connection to access the application and interact with the database.

5. Non-functional Requirements

5.1 Performance Requirements

- **Speed**: The system should respond to user actions (like logging in or performing transactions) within 5 seconds.
- **Availability**: The application should be available for use 24/7, with minimal downtime.

5.2 Safety Requirements

- **Data Backup**: The system should regularly save data to avoid losing important information.
- **Recovery**: If the system crashes, it should be able to restore data without losing important details like account balances or transaction history.

5.3 Security Requirements

- **Password Protection**: Users must log in with a password to access their accounts.

- **Data Protection**: Basic security should be implemented to protect sensitive data, like account information.

5.4 Software Quality Attributes

- **Usability**: The system should be easy to use, with simple screens and clear instructions.
- **Reliability**: The application should work properly without crashing or showing errors frequently.