Automated Banking System - Software Requirements Specification (SRS)

1. Problem Statement

The Automated Banking System aims to provide customers with a seamless digital banking experience, automating core operations such as account management, money transfer, and transaction tracking.

2. Background

As technology advances, the banking sector is striving to enhance its services through automation and digitalization. This system leverages Java Full Stack technologies to streamline banking operations, ensuring accuracy, speed, and customer satisfaction.

3. Problem Description

Traditional banking processes are manual, time-consuming, and error-prone. This system automates key operations, maintaining accurate balances, recording audit logs, and offering advanced features like ACID-compliant SQL operations, DynamoDB mirroring, stack/queue transaction handling, BDD testing, and DevOps automation.

4. Scope

The system will provide: 1. Customer Management – Register new customers, authenticate login, and manage credentials. 2. Account Management – Create accounts, view balances, close accounts, change passwords. 3. Transaction Management – Deposit, Withdraw, Transfer (intra-bank), Transaction History. 4. Audit Logs – Record every transaction with before/after balances. 5. Advanced Features – ACID-safe SQL operations, DynamoDB log mirroring, Data Structures (stack/queue), BDD testing, cron job automation.

5. Functional Requirements

5.1 Customer Management

- Register (Sign Up): Full name, email, password with validation.
- Login: Email + password authentication.
- Change Password: Update password.

5.2 Account Management

- Create Account: Type (SB/CA), initial deposit (+ve integer).
- View Balance: Show all accounts and balance for selected account.
- Close Account: Remove account.

5.3 Transaction Management

- Deposit Money
- Withdraw Money (with success/fail validation)
- Transfer Money (intra-bank, with validation)
- Transaction History: List past transactions.

5.4 Audit Logs

- Automatically generate logs for Deposit, Withdraw, Transfer.
- Fields: txn_id, account_number, actor, action, before_balance, after_balance, timestamp.
- Immutable.

5.5 Advanced Features

- **SQL/ACID:** Atomic & isolated balance updates.
- **DynamoDB:** Mirror audit logs for eventual consistency.
- **Data Structures:** Use stack and queue for transaction handling and retries.
- **BDD (Cucumber):** Test transfer scenario.
- **DevOps (cron jobs):** Daily settlement job.

6. Non-Functional Requirements

- **Consistency:** Accurate balances.
- **Atomicity:** Transactions fully succeed or fail.
- **Security:** Validate customer credentials.
- **Traceability:** Full audit trail.
- **Durability:** Data persists reliably.

7. User Stories

7.1 Customer Registration

- Input: Full Name, Email, Password.
- Output: Customer created successfully, thank you message.

7.2 Customer Login

- Input: Email, Password.
- Output: Menu with Create Account, View Balance, Withdraw, Transfer, Transaction History, Close Account, Change Password, Exit.

7.3 Create Account

- Input: Account type (SB/CA), Initial deposit.
- Output: Account created successfully.

7.4 View Balance

- Input: Select account.
- Output: Show balance.

7.5 Withdraw Money

- Success: Amount <= balance → Withdraw successful, updated balance.
- Fail: Amount > balance → Show error, balance unchanged.

7.6 Transfer Money

- Input: Source & destination accounts, amount.
- Validation: Amount <= source balance, destination account valid.
- Output: Amount transferred successfully, audit logs created.

7.7 Transaction History

- Input: Select account.
- Output: List of past transactions.

8. Notes

- Fully customer-centric, no employee/admin functionalities.
- Includes advanced features like ACID SQL, DynamoDB, DS, BDD, and cron job automation.

End of SRS Document