Software Requirements Specification (SRS) for Corporate Banking Payment Application

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the corporate banking application. This document outlines the functional and non-functional requirements, the system architecture, and the various user roles and their permissions.

1.2 Scope

The corporate banking application is designed to facilitate banking operations for corporate clients, banks, and super administrators. The application will include functionalities such as customer onboarding, client management, payment processing, salary disbursement, and report generation. The project will be developed using Spring Boot for the backend and Angular for the frontend.

1.3 Definitions, Acronyms, and Abbreviations

- **CRUD**: Create, Read, Update, Delete
- **SRS**: Software Requirements Specification
- **OAuth**: Open Authorization
- **CAPTCHA**: Completely Automated Public Turing test to tell Computers and Humans Apart
- **API**: Application Programming Interface

2. Overall Description

2.1 Product Perspective

The corporate banking application will be a web-based system accessible through standard web browsers. It will interact with existing banking systems via APIs and will be integrated with secure authentication mechanisms such as OAuth.

2.2 Product Functions

The application will support the following primary functions:

- User authentication and authorization
- Bank and client management
- Customer onboarding and verification
- Payment processing and approval
- Salary disbursement to employees
- Document upload and management
- Batch processing of transactions
- Report generation

2.3 User Classes and Characteristics

- **Super Admin**: Manages banks and generates various reports.
- **Bank User**: Manages clients, verifies customers, approves/rejects payments, uploads documents, and generates reports.
- **Client User**: Manages beneficiaries and employees, processes payments, disburses salaries, and views reports.

2.4 Operating Environment

- Backend: .Net Core
- Frontend: Angular
- Database: SQL Server
- Authentication: OAuth 2.0 (JWT)
- Browser Compatibility: Chrome, Firefox, Edge, Safari

3. System Features

3.1 Super Admin Features

- **Manage Banks**: Super Admin can create, read, update, and delete bank records.
- **Generate Reports**: Super Admin can generate and view various reports, including system usage and audit logs.

3.2 Bank User Features

- **Customer Onboarding**: Bank users can onboard new customers by entering their details and uploading necessary documents.
- **Manage Clients**: CRUD operations on client records and verification of client information.
- **Approve/Reject Payments**: Bank users can approve or reject payment requests from clients.
- **Transaction Reports**: Generate transaction reports for each customer.
- **Document Uploads**: Upload and manage documents related to clients and transactions.

3.3 Client User Features

- **Manage Beneficiaries**: Clients can create, read, update, and delete beneficiaries.
- **Manage Employees**: Clients can manage employee records, including details necessary for salary disbursement.
- **Process Payments**: Clients can make payments to other clients.
- **Salary Disbursement**: Clients can disburse salaries to their employees, either individually or in batch for all employees.
- **Generate Reports**: Clients can generate and view reports related to their transactions, payments, and salary disbursements.

4. External Interface Requirements

4.1 User Interfaces

The application will provide intuitive and responsive web interfaces for all user roles, ensuring ease of use and accessibility.

4.2 APIs

The system will expose RESTful APIs for various functionalities such as customer onboarding, payment processing, and report generation.

4.3 Authentication

The application will use OAuth 2.0 for secure authentication and authorization.

5. System Requirements

5.1 Functional Requirements

- **FR1**: The system shall allow Super Admin to manage bank records (CRUD operations).
- **FR2**: The system shall allow Bank Users to onboard customers, including document uploads.
- **FR3**: The system shall allow Bank Users to manage client records (CRUD operations) and verify clients.
- **FR4**: The system shall allow Bank Users to approve or reject payment requests.
- **FR5**: The system shall allow Bank Users to generate transaction reports for each customer.
- **FR6**: The system shall allow Client Users to manage beneficiaries (CRUD operations).
- **FR7**: The system shall allow Client Users to manage employee records, including necessary details for salary disbursement.
- **FR8**: The system shall allow Client Users to make payments to other clients.
- **FR9**: The system shall allow Client Users to disburse salaries to their employees, either individually or in batch for all employees.
- **FR10**: The system shall allow Client Users to generate transaction, payment, and salary disbursement reports.
- **FR11**: The system shall support document uploads and management for Bank and Client Users.
- **FR12**: The system shall support batch processing of transactions.
- **FR13**: The system shall implement CAPTCHA for form submissions to prevent automated attacks.

5.2 Non-Functional Requirements

- **NFR1**: The system shall ensure data security and privacy through encryption and secure communication channels.
- **NFR2**: The system shall provide high availability and reliability with an uptime of 99.9%.
- **NFR3**: The system shall be scalable to handle increasing loads and user counts.

- **NFR4**: The system shall ensure performance efficiency, with response times not exceeding 2 seconds for any operation.
- **NFR5**: The system shall be maintainable and support easy updates and bug fixes.

6. Security Requirements

- **SR1**: The system shall use OAuth 2.0 for secure authentication and authorization.
- **SR2**: The system shall implement CAPTCHA to prevent automated form submissions.
- **SR3**: The system shall encrypt sensitive data both at rest and in transit.
- **SR4**: The system shall log all access and modification activities for audit purposes.

7. Project Management Requirements

7.1 Development Environment

- **Backend**: .NetCore
- **Frontend**: Angular
- **Database**: SQL Server

7.2 Tools and Technologies

- **Version Control**: Git

8. Glossary

- **Super Admin**: A user role with the highest level of privileges, responsible for managing banks and generating reports.
- **Bank User**: A user role responsible for client management, payment approvals, and report generation within a specific bank.
- **Client User**: A user role responsible for managing beneficiaries, employees, payments, and salary disbursements.

9. System Flow

9.1 Overview

The system flow describes how users interact with the corporate banking application and how various components communicate with each other to fulfil the application’s requirements. The following sections provide a detailed flow for each major functionality, including user authentication, customer onboarding, client management, payment processing, salary disbursement, document uploads, and report generation.

9.2 User Authentication and Authorization

1. **User Login**
    - User accesses the login page.
    - User enters credentials (username and password).

    - System validates credentials using OAuth 2.0.(JWT)
    - If credentials are valid, the system generates a secure token and redirects the user to their dashboard.
    - If credentials are invalid, the system displays an error message.

9.3 Customer Onboarding

1. **Initiate Onboarding**
    - Bank user logs into the system.
    - Bank user navigates to the "Customer Onboarding" section.
    - Bank user enters customer details (e.g., name, address, contact information).
2. **Document Upload**
    - Bank user uploads required documents (e.g., ID proof, address proof).
    - System validates and stores the documents securely.
3. **Verification**
    - Bank user submits the customer details and documents for verification.
    - System sends a notification to the verification team.
    - Verification team reviews the submitted details and documents.
    - Verification team approves or rejects the customer onboarding request.
    - System notifies the bank user of the approval/rejection status.

9.4 Client Management

1. **Manage Clients**
    - Bank user logs into the system.
    - Bank user navigates to the "Client Management" section.
    - Bank user performs CRUD operations on client records.
    - System updates the client records in the database.
2. **Client Verification**
    - Bank user verifies the client information.
    - System updates the verification status of the client.

9.5 Payment Processing

1. **Initiate Payment**
    - Client user logs into the system.
    - Client user navigates to the "Payments" section.

    - Client user selects the beneficiary and enters payment details.

2. **Payment Approval**
    - Payment request is submitted for bank user approval.
    - Bank user reviews and approves/rejects the payment request.
    - System processes the payment upon approval.
    - System notifies the client user of the payment status.

9.6 Salary Disbursement

1. **Manage Employees**
    - Client user logs into the system.
    - Client user navigates to the "Employee Management" section.
    - Client user performs CRUD operations on employee records.
2. **Disburse Salaries**
    - Client user navigates to the "Salary Disbursement" section.
    - Client user selects employees and enters salary details.
    - Client user can choose to disburse salaries individually or in batch.
    - System processes the salary disbursements.
    - System generates notifications and receipts for salary transactions.

9.7 Document Uploads

1. **Upload Documents**
    - Bank or client user logs into the system.
    - User navigates to the relevant section for document uploads.
    - User uploads documents related to customers, clients, or transactions.
    - System validates and stores the documents securely.
2. **Access Documents**
    - Authorized users can access and download uploaded documents.
    - System ensures secure access based on user roles and permissions.

9.8 Report Generation

1. **Generate Reports**
    - User logs into the system.
    - User navigates to the "Reports" section.

- User selects the type of report to generate (e.g., transaction reports, payment reports, salary disbursement reports).
- User specifies any filters or parameters for the report.

2. **View and Download Reports**
   - System generates the report based on the specified criteria.
   - User views the report on the screen.
   - User can download the report in various formats (e.g., PDF, Excel).

Cloudinary

API Key : 212444375392957

API Secret - uvNVmNRInTp5UU0BbhL9KqbywJI

Cloud_name – dtwilhjhw

CLOUDINARY_URL=cloudinary://<your_api_key>:<your_api_secret>@dtwilhjhw

Cloudinary Image Manipulation - https://cloudinary.com/documentation/dotnet_image_manipulation

Cloudinary Sample Projects - https://github.com/cloudinary/CloudinaryDotNet/tree/master/samples

Cloudinary Qick Start - https://cloudinary.com/documentation/dotnet_quickstart

```
⇨ Create a .env file with Environment Variable OR  An Account class can also be created
⇨ var account = new Account(
⇨    cloud: "your_cloud_name",
⇨    apiKey: "your_api_key",
⇨    apiSecret: "your_api_secret"
⇨ );
⇨
⇨ var cloudinary = new Cloudinary(account) // URL gets created automatically??
⇨ {
⇨    Api = { Secure = true }
⇨ };
```

Delete Files from Coudinary

Deleting Uploaded files from Cloudinary with ajax

JQGrid Display

https://www.c-sharpcorner.com/article/using-jqgrid-with-asp-net-mvc/

https://free-jqgrid.github.io/getting-started/index.html

colModel options => http://www.trirand.com/jqgridwiki/doku.php?id=wiki:colmodel_options https://www.guriddo.net/documentation/guriddo/javascript/user-guide/formatters/#formatter-select

Editing => https://www.guriddo.net/documentation/guriddo/javascript/user-guide/editing/

CRUD Operations => Performing CRUD Operations Using jqGrid In ASP.NET MVC

General Info => http://www.trirand.com/jqgridwiki/doku.php?id=wiki:colmodel_options

Note that the JQuery – js and css have to be in proper sequence and compatible versions for the jqGrid to work

Sorting, Searching & Pagination in MVC

https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/sorting-filtering-and-paging-with-the-entity-framework-in-an-asp-net-mvc-application

Charts

https://www.codeguru.com/dotnet/creating-different-types-of-charts-in-asp-net-mvc/

https://canvasjs.com/docs/charts/integration/asp-net-mvc-charts/