Software Requirements Specification (SRS)

Version: 1.0

1. Introduction

1.1 Purpose

This document describes the functional and non-functional requirements for the Online Virtual Card Platform. The purpose of this system is to provide a secure, scalable, and compliant platform for users to register, complete identity verification (KYC), request and manage virtual payment cards, and perform transactions. This SRS will serve as a reference for project managers, developers, testers, and stakeholders throughout the project lifecycle.

1.2 Project Scope

The platform will be a web and mobile application enabling end-users to manage virtual cards. The system will handle user registration, secure authentication, KYC processes, integration with third-party card issuers for card lifecycle management, and transaction processing. Administrative users will have interfaces for user, KYC, and card oversight.

1.3 Definitions, Acronyms, and Abbreviations

- FR: Functional requirements.
- NFR: Non-functional requirements.

1.4 References

- system design.pdf

2. Overall Description

2.1 Product Perspective

This system is a self-contained product but will integrate with several external third-party services, including:

- Card Issuing Provider (e.g., Stripe Issuing)
- Payment Processors (e.g., MTN, ORANGE)
- Identity Verification Services for KYC

2.2 Product Functions

The core functions of the system are:

- User Registration and Profile Management
- Secure User Authentication and Authorization
- KYC Verification and Status Tracking

- Virtual Card Issuance, Freezing, Unfreezing, and Closure
- Transaction Recording, History, and Dispute Initiation
- Real-time User Notifications

2.3 User Classes and Characteristics

- **End-User:** Registers, completes KYC, requests and uses virtual cards.
- **Administrator:** Manages platform operations, has full system access.
- **Compliance Officer:** Manages the KYC manual review queue and approvals/rejections.

2.4 Assumptions and Dependencies

- The system is dependent on the availability and API stability of the third-party Card Provider and KYC verification services.
- It is assumed that users have access to a smartphone or computer with a modern web browser.

TECHNOLOGY STACK

- **Frontend:** React.js for web, React Native for mobile apps.
- **Backend:** Java-based application server (e.g., Spring Boot).
- **Database:** PostgreSQL for persistent data storage.
- **Cache:** Redis or similar for frequent request caching.
- **Infrastructure:** Cloud-based (AWS, Azure, GCP) for scalability and availability.

3. Specific Requirements

3.1 Functional requirements

3.1.1 User Management

- The system shall allow a new user to create an account by providing necessary details.
- The system shall allow users to view and update their own profile information.
- The system shall allow administrators to view, update, and deactivate user accounts.
- The system shall assign and manage user roles (User, Admin).

3.1.2 User Authentication

- The system shall provide a secure login mechanism using email and password.
- The system shall provide a "Forgot Password" flow for secure password reset.
- The system shall support OAuth2.0 for social login .
- The system shall enforce role-based access control for all endpoints.

3.1.3 KYC (Know Your Customer) Management

- The system shall allow users to upload identity documents (e.g., ID card, passport).
- The system shall integrate with a third-party service to perform automated identity checks.
- The system shall track and display the KYC status (Pending, Approved, Rejected) to the user.

3.1.4 Virtual Card Management

- The system shall allow KYC-verified users to request the issuance of a new virtual card via a 3rd party card provider.
- The system shall display card metadata (last 4 digits, expiry date) to the user.
- The system shall allow users to freeze and unfreeze their virtual cards.
- The system shall allow users to permanently close their virtual cards.
- The system shall support setting spending limits and controls per card.

3.1.5 Transaction Management

- The system shall receive transaction events from the card provider.
- The system shall record and store all transaction details.
- The system shall provide users with a transaction history.

3.1.6 Notification Service

- The system shall notify users via email or SMS for critical events (e.g., KYC status change, card transaction, security alert).

3.2 External Interface Requirements

3.2.1 User Interfaces

- A responsive web application built with React and Tailwind CSS.
- A mobile application for iOS and Android built with React Native.

3.2.2 Software Interfaces

- **Card Provider API:** For card issuance, suspension, and receiving transaction webhooks.
- **KYC Provider API:** For automated identity document verification and background checks.
- **Payment Provider API:** For facilitating top-ups and funding transactions.

3.2.3 Communications Interfaces

- The system shall use HTTPS for all client-server and server-to-server communications.
- The system shall expose a RESTful API for the frontend clients.
- The system shall consume webhooks from the Card Provider over HTTPS.

3.3 Non-Functional Requirements

3.3.1 Security Requirements

- All sensitive data at rest (e.g., in the database) shall be encrypted.
- All user passwords shall be hashed and salted before storage.
- The system shall implement robust session management and protection against common web vulnerabilities.

3.3.2 Performance Requirements

- The system's API shall respond to 95% of typical user requests.
- The system shall be designed to support thousands of daily transactions and a growing user base.

3.3.3 Software Quality Attributes

- **Availability:** The system shall maintain 99.9% uptime to ensure users can perform transactions at any time.
- **Scalability:** The system architecture shall be scalable to handle peak loads without degradation of performance.
- **Compatibility:** The platform shall function correctly on the latest versions of major web browsers (Chrome, Firefox, Safari, Edge) and mobile operating systems (iOS, Android).
- **Maintainability:** The code-base shall be well-documented and structured to facilitate easy updates and feature additions.