Name: Mambo Annabel & Fru Patience

Department: Software Engineering

Campus: B

SYSTEM ARCHITECTURE DESIGN VIRTUAL CARD FINANCIAL SYSTEM

Executive Summary

This architecture document outlines the design of a scalable, secure, and modular financial platform that enables users to create and manage virtual payment cards, perform digital transactions, and maintain wallet balances through a mobile application. The system is designed around a microservices architecture, which ensures scalability, fault tolerance, and independent service deployment.

The system integrates advanced fraud detection mechanisms, KYC verification, and real-time payment processing, while adhering strictly to PCI-DSS (Payment Card Industry Data Security Standard) compliance. Security is treated as a core pillar, with tokenization mechanisms in place to protect sensitive financial data.

The intended client application is a Flutter-based mobile app, which connects securely to the backend via an API Gateway. The backend infrastructure is containerized, ensuring high availability and elasticity in response to transaction volume.

Technology Stack

The technology stack is carefully selected to balance **performance**, **security**, and **developer productivity**:

- **Mobile App:**
  The mobile interface is built with Flutter, offering a seamless and high-performance experience across Android and iOS. Flutter's reactive framework

allows for real-time updates (such as transaction status or wallet changes) through API responses and push notifications.

- **Backend:**
  Go is chosen for its concurrency efficiency and speed, especially for payment processing and authentication services. Python complements Go by powering machine-learning-based fraud detection models due to its mature AI/ML ecosystem.
- **API Gateway:**
  Acts as the entry point for all external requests, handling **load balancing**, **rate limiting**, and **routing** of requests to the appropriate microservices. NGINX also provides **SSL/TLS termination** and protects against DDoS and brute-force attacks.
- **Databases:**
  PostgreSQL serves as the primary relational database for transactional data, ensuring ACID compliance and strong data consistency. Redis is used for in-memory caching**,** session management**,** and temporary OTP/MFA storage to boost performance.
- **Messaging Queue:**
  Kafka ensures **asynchronous communication** between microservices, enabling real-time event-driven workflows such as fraud alerts, payment events, and notification triggers. This decoupling improves reliability and system scalability.
- **Deployment & Infrastructure:**
  - **Docker** packages each microservice into lightweight containers for portability.
  - **Kubernetes** automates scaling, health checks, and self-healing deployments.
  - **Terraform** manages infrastructure as code (IaC), making environment setup consistent across staging and production.
- **Security Layer:**
  Sensitive data such as encryption keys, API credentials, and tokens are stored securely in HashiCorp Vault and managed through Key Management Services

(KMS)**.** Hardware Security Modules (HSMs) ensure the physical protection of encryption keys. TLS 1.3 secures all communication channels.

- **Monitoring & Observability:**
  Prometheus collects real-time metrics from microservices, while Grafana visualizes performance trends and system health. This allows proactive detection of bottlenecks and early identification of potential failures.

CORE MODULES

1. Authentication & Authorization (

Handles all user identity-related functions. The system implements OAuth2 for secure authorization and uses JWTs (JSON Web Tokens) for stateless session management. Multi-Factor Authentication (MFA) strengthens access control by requiring verification via SMS, email, or authenticator apps.

2. User Management Service

Responsible for creating, updating, and managing user profiles. It maintains user preferences, account limits, and device binding information. Data integrity is enforced through strict schema validation and audit logging.

3. KYC Verification Service

Implements customer identity checks to comply with financial regulations. This service integrates with third-party verification providers (e.g., Onfido, Trulioo) to perform ID scans, facial recognition, and document verification.

Workflow Example:
User uploads their ID → Service validates with provider → Verification result is saved in user profile → KYC status determines transaction permissions.

4. Rule Engine (Fraud Detection)

A specialized microservice powered by Python-based AI models. It continuously analyzes user transaction patterns, device fingerprints, geolocation, and behavioral data to detect potential fraud.

Example Rules:

- Block payments exceeding a threshold from new devices.
- Flag rapid successive withdrawals from different IP addresses.
- Use ML models to detect abnormal transaction clusters.

Alerts generated here are forwarded asynchronously via Kafka to the **Security Operations Center (SOC)** or manual review team.

5. Payment Processing Service (3rd-Party Integration)

This is the heart of the financial system. It connects to **external payment processors**, **banks**, and **card networks (e.g., Visa, MasterCard)** to initiate and settle transactions.

Process Overview:
When a user makes a payment, the service constructs an ISO 8583 or REST-based message, sends it to the payment processor, and waits for a synchronous response (approved, declined, or pending). It also handles **currency conversion**, **transaction fee calculation**, and **ledger updates**.

6. Notification Service (Email, SMS, Push)

This service handles all communication to the user. It supports multiple channels including **email notifications**, **SMS alerts**, and **mobile push notifications** via Firebase or APNs.

Example Use Cases:

- Notify user when a card is created or topped up.

- Send alerts for failed or suspicious transactions.
- Push wallet balance updates after payments.

Kafka ensures these messages are queued and delivered reliably even under high load.

7. Wallet & Cashing Service

Manages user wallets, including balance tracking**,** fund transfers, and virtual card **issuance**. Every transaction updates a ledger table to ensure full traceability for audits. It integrates tightly with the Payment Service and Notification Service**.**

Functions Include:

- Top-up wallet via bank or card.
- Transfer between wallets.
- Update available balance after payments or refunds.
- Synchronize with the payment processor's settlement reports.

REQUEST–RESPONSE FLOW

Below is the typical flow for a user-initiated transaction**:**

1. **User Login**
   The user logs in from the Flutter app → request passes through NGINX API Gateway → Authentication Service validates credentials and issues a **JWT** token.
2. **Transaction Initiation**
   User initiates a payment request → forwarded to the Backend-for-Frontend (BFF) service → verifies user identity and checks **KYC status**.
3. **Fraud Check**
   BFF calls the **Rules Engine** → evaluates transaction risk → either proceeds or flags for review.
4. **Payment Processing**
   The **Payment Connector** sends transaction details to the **third-party payment processor** → waits for authorization response (approved/declined).

5. **Notification Trigger**

   Based on the transaction result, the **Notification Service** dispatches SMS, email, and/or push messages to inform the user of the status.

6. **Wallet Update**

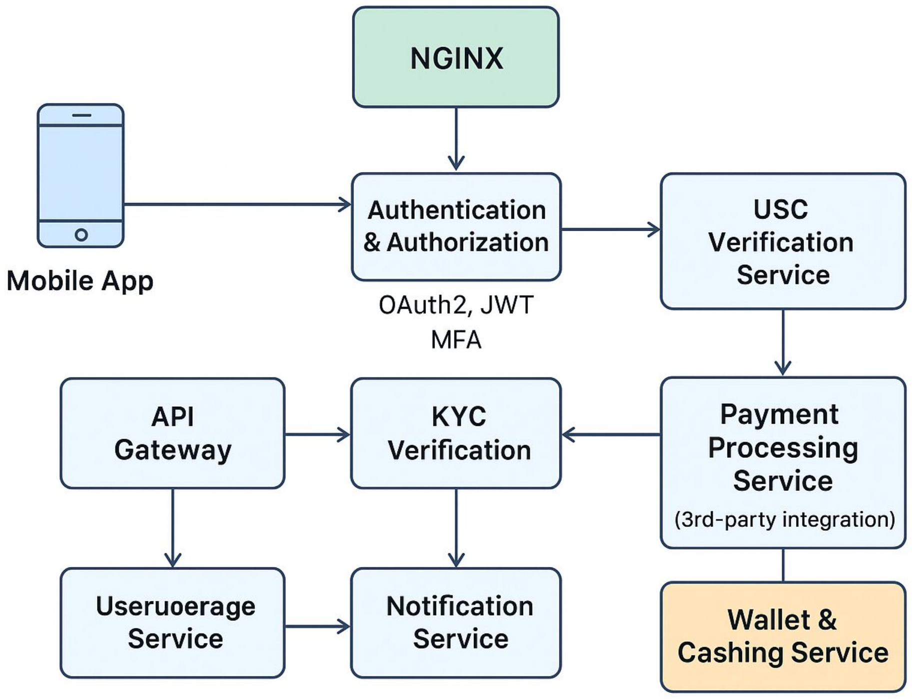
   The **Wallet Service** updates the user's wallet balance and records the ledger entry for traceability.

7. **Response Delivery**

   API Gateway returns a structured response back to the mobile app, completing the transaction cycle.

- **Audit Trails:**

  Every action whether user-initiated or system-generated—is logged with metadata (timestamp, actor, IP, event type). This supports compliance audits and incident investigations.


NGINX
Mobile App
Authentication & Authorization
OAuth2, JWT
MFA
USC Verification Service
API Gateway
KYC Verification
Payment Processing Service
(3rd-party integration)
Useruoerage Service
Notification Service
Wallet & Cashing Service