System Design Mock Project: Payment Gateway Backend

Overview

This mock system design project outlines the backend architecture for a Payment Gateway (like Razorpay or PayPal). It simulates a real-world high-level system, including component diagrams, API design, database schema, and scalability considerations.

Objective

To design a secure, scalable, and modular backend system that can:

- Accept and process online payments
- Interface with external banking APIs
- Handle refunds and transaction queries
- Log and store all transaction data

Functional Requirements

- Accept payment requests via API
- Validate merchant/user credentials
- Process payments and generate transaction IDs
- Interact with external mock bank system
- Allow refund/cancel transaction API
- Store all transaction details securely

Non-Functional Requirements

- High availability and scalability
- Secure PIN/token-based authentication
- Logging and monitoring support
- Rate limiting for public APIs

- Fault-tolerant (retry on failure)

System Components

- API Gateway
- Payment Service
- User Service
- Transaction Service
- Bank Adapter/Integration Layer
- Database

Database Schema (Simplified)

- Users Table: id, name, email, token, wallet_balance
- Merchants Table: id, name, key, status
- Transactions Table: id, user_id, merchant_id, amount, status, timestamp
- Logs Table: id, message, level, timestamp

Sequence Flow

1. User/merchant calls POST /pay
2. API Gateway validates API key/token
3. Payment Service receives request
4. Calls User Service to check balance
5. Calls Bank Adapter to charge card or bank
6. Transaction Service logs transaction and returns ID
7. Returns success/failure JSON

API Endpoints

POST /pay

- Input: userToken, merchantId, amount

- Output: transactionId, status

POST /refund

- Input: transactionId

- Output: refundStatus

GET /status/:transactionId

- Output: current status of payment

Security

- Token-based authentication

- Input validation

- Rate limiting

- Encryption

Logging and Monitoring

- Log requests/responses with timestamp

- Levels: Info, Warn, Error

Scalability

- Stateless APIs for scaling

- Async retry queue

- Load balancer

- CDN caching for GET requests

Deliverables

- System Design PDF (this file)

- Optional: CLI Simulation

- Architecture Diagram (Draw.io/Lucidchart)

- API Markdown documentation

Use Cases

- Fintech/backend interview prep

- Portfolio system design example

- Resume project for system-oriented roles

Author: [Your Name]