System design overview – virtual payment platform

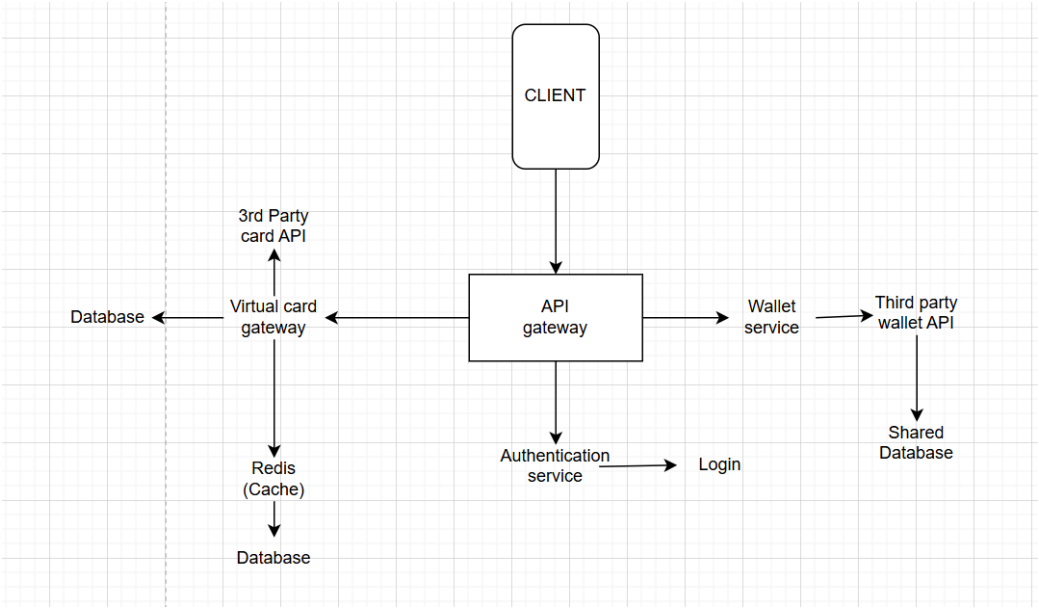
This document provides an overview of the system design for a virtual payment platform developed by Company XYZ. The platform enables users to make digital payments and manage virtual cards through a secure and scalable system. It integrates several modules, including Authentication, Wallet Management, Virtual Card API integration, and Transaction Management using Redis for caching.

1. System Architecture Summary

The system follows a client-server model with a modular microservice architecture. Each service handles specific functionalities such as authentication, card operations, and wallet management. The system is designed for high scalability, reliability, and performance.

2. Core System Components

1. Client Layer: Web and mobile interfaces for user interaction.
2. API Gateway: Manages all client requests and routes them to the correct backend service.
3. Authentication Service: Handles user login, signup, and token-based authentication.
4. Wallet Service: Manages deposits, withdrawals, and balance inquiries.
5. Virtual Card Service: Integrates with third-party APIs (e.g., Stripe, Paystack, Flutterwave) for card creation and funding.
6. Transaction Service: Logs and monitors all financial activities.
7. Redis Cache Layer: Speeds up performance by caching session data and transaction histories.
8. Database Layer: Stores persistent data (users, cards, transactions, etc.).

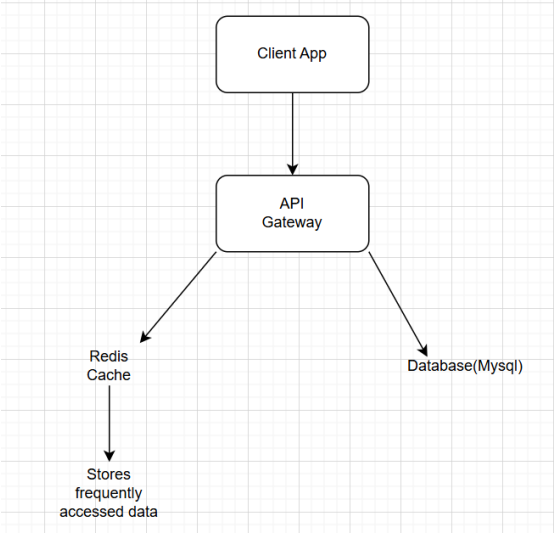


3. Redis in Data Caching

Redis (Remote Dictonary Server) is an in memory key-value data store used to cache frequently accessed data so that the system doesn't always have to query the from the main database. It is extremely fast given that is stores data in RAM instead of on a disk.

Redis helps to:

- Speed up performance: reduces delay when users carries our queries like view balance
- Reduces load on the main database
- Maintain temporary transactions data: Redis can store pending payments before they are written permanently to the database. When new transactions occur, Redis temporarily holds transactions data before it's confirmed and saved performance.



4. Recommended Technologies

9. Frontend: React Native / Flutter
10. Backend: Node.js (Express) or Python (Flask/FastAPI)
11. Database: PostgreSQL or MySQL
12. Cache: Redis
13. API Gateway: Nginx / Express Gateway
14. 3rd Party APIs: Stripe, Paystack, Flutterwave

5. Deliverables

15. Client-Server Architecture Diagram (Draw.io/Figma).
16. Redis Caching Architecture Diagram.
17. Overall System Architecture Diagram.
18. Technical Report and PDF documentation.

This design ensures modularity, scalability, and data security. Redis caching improves system performance, while API integration with third-party services enables real-time virtual card processing. The architecture supports future extensions and enhanced analytics capabilities.