Banking Microservices Application

Figma file Link **(scan this link and paste in browser)**

https://www.figma.com/design/NPQHpglfPVe8LSZti349ZO/Banking-App--Community-?node-id=0-1&p=f&t=vZ66RW6pyrhSeUBU-0

Note:-

If you find this project useful, you can use it in your own projects and add any additional features you need. If I've missed something, feel free to extend or modify it as required

Banking APP


Banking
Your Cards
2 pysical card, 1 virtual card
VISA
3.469.00 USD
14.590 USD
$4153.93
Scan QR Code

CONTENTS

1. Problem Statement

Build a secure, scalable Mobile banking platform that supports:

- Customer onboarding & authentication
- Account management (savings/current)
- Money transfers (internal/external)
- Transaction history & statements
- Notifications (SMS/Email)
- Audit logging & reporting

Goals

- High reliability and data consistency for monetary operations
- Clear separation of domains (microservices)
- Observability, security, and compliance from day one

Work Flows

2.1 Customer Onboarding (Sign-up + KYC)

1. **Auth Service**: Create user → send verification OTP
2. **Customer Service**: Save profile + KYC docs (status = pending)
3. **Compliance Bot** (optional async): Approves/Rejects KYC
4. **Notification Service**: Send result

2.2 Login & Session

1. **Auth Service**: Verify credentials/MFA → issue JWT (short) + Refresh token (long)
2. **API Gateway**: Validates JWT on each request

2.3 Open Account

1. **Account Service**: Create account (IBAN/AccNo), initial balance = 0
2. **Ledger Event**: `ACCOUNT_OPENED` → **Audit Service** logs

2.4 Internal Transfer (A → B, same bank) — Saga

1. Client → **Transaction Service** `POST /transfers` (with idempotency key)
2. **Transaction Service** (Orchestrator):
   - Debit A (reserve funds) via **Account Service**
   - Credit B via **Account Service**
   - Mark transaction `COMPLETED` or run **compensation** on failure (unreserve)
3. Publish `TRANSFER_COMPLETED` → **Notification** + **Audit**

2.5 External Transfer (NEFT/RTGS/UPI placeholder)

1. Client → **Transaction Service**
2. Debit local account
3. Send payment message to **Payments Service** (connector to external rails)
4. On success/fail callback → finalize & notify

2.6 Transaction History / Statement

1. Client → **Transaction Service** `GET /transactions?accountId=…&page=…&size=…&sort=…&from=…&to=…`
2. Returns **paginated** list with running balance (or compute on UI)

2.7 Notifications

- Subscribe to domain events: `TRANSFER_COMPLETED`, `LOW_BALANCE`, `KYC_APPROVED`
- Send SMS/Email; store templates & delivery logs

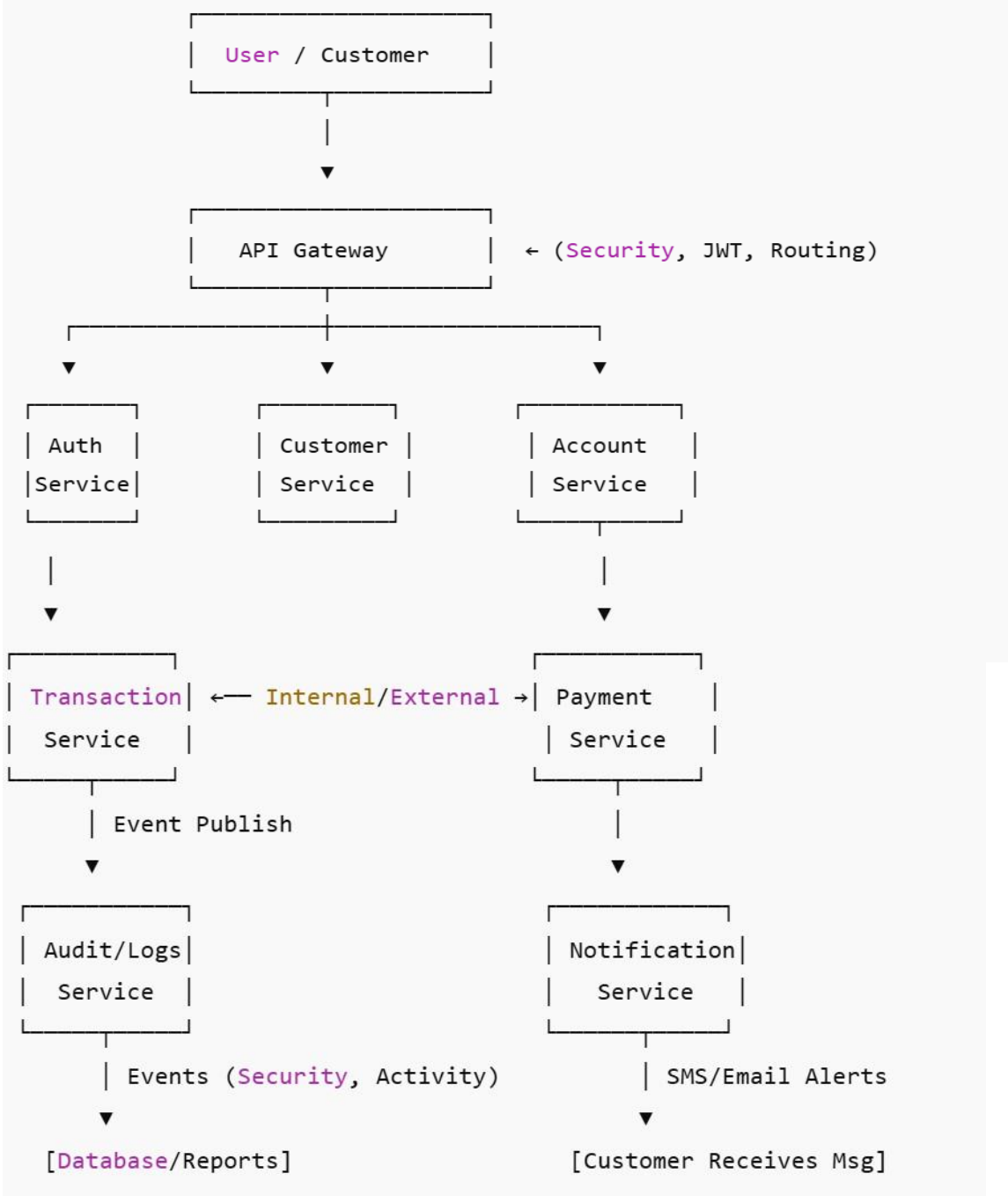
3. Technical Details (incl. Best Practices)

3.1 Services (own DB each)

- **Auth Service**: JWT/OAuth2, MFA/OTP, refresh tokens
- **Customer Service**: Profiles, KYC, contact info
- **Account Service**: Accounts, balances, holds, statements (read model)
- **Transaction Service**: Orchestrates transfers (Saga), idempotency
- **Payments Service** (ext integration): NEFT/RTGS/UPI mock
- **Notification Service**: SMS/Email
- **Audit Service**: Immutable audit logs
- **Reporting Service** (later): Aggregations, BI

> **Best practice:** Database-per-service; avoid cross-service joins. Communicate via APIs/events.

🏦 Overall Flow



✅ Outcome of This Project

By the end of this project, you'll have a **modular, production-style banking system** that mimics real-world banks.

🔹 Core Features Delivered

1. **User Management & Authentication**
   - Secure login with JWT
   - Customer onboarding with KYC
2. **Account Management**
   - Create Savings/Checking accounts
   - Debit, Credit, Balance tracking
3. **Transactions**
   - Internal transfers (A → B within same bank)
   - External payments (NEFT/UPI simulation)
   - Transaction history with **pagination + filters**
4. **Notifications**
   - SMS/Email for transfers, low balance, account creation

5. **Auditing & Logging (AOP best practice)**
    - Every transfer, login, and account event logged
6. **Scalability with Microservices**
    - Independent services (Auth, Accounts, Payments, Notifications)
    - Easy to extend (add Loans, Cards later)

Documentation Using API Doc's and Swagger UI

(refer this for swagger UI & Api doc's
---->https://github.com/SuryaBhaskarG/rest-semi
----->https://github.com/thevipulvats/journalApp

Tech Stack Overview with Icons

Below is your refined tech stack summary—each paired with the relevant icon above:

1. **Spring Boot** – Backend services framework
2. **Angular** – Frontend SPA framework
3. **MySQL** – Relational database for core transactional data
4. **MongoDB** – NoSQL database for audit logs or unstructured data
5. **Apache Kafka** – Stream/event messaging backbone
6. **Bootstrap** – Frontend UI styling + responsive components (via Angular)
7. **JWT / OAuth2** – Security protocols for authentication & authorization