Commerce Bank App – System Design Document

1. Purpose of the Document

The purpose of this document is to provide a clear overview of the Commerce Bank application's architecture, functionality, and technology choices. It outlines the system design, technical requirements, and major architectural decisions to ensure maintainability, scalability, and security throughout the development process.

2. System Overview

Problem Statement

In today's digital era, users expect a seamless, secure, and efficient way to manage their banking activities online. Traditional banking systems often suffer from slow processing times, lack of personalization, and limited accessibility.
The **Commerce Bank App** aims to solve these issues by providing an intuitive and reliable web-based banking platform where users can securely manage their accounts, transfer funds, and track transactions in real-time.

User Types

1. **Customers:**
    - Can create and manage accounts.
    - View balance and transaction history.
    - Perform money transfers securely.
2. **Bank Admins:**
    - Manage user accounts.
    - Approve transactions or flag suspicious activity.
    - Generate reports on overall system activity.

Key Features

- **User Authentication & Authorization** (JWT-based login system).
- **Account Management:** Create, update, and delete accounts.
- **Fund Transfer:** Send and receive money between users.
- **Transaction History:** View past transactions with timestamps.

- **Admin Dashboard:** Manage users and monitor system activity.

3. Functional and Non-Functional Requirements

Functional Requirements

- Users must be able to **register, log in, and log out** securely.
- System should **validate user credentials** before granting access.
- Users can **view account details and transaction history**.
- Users can **transfer money** between accounts.
- Admins can **view all users** and **manage transactions**.

Non-Functional Requirements

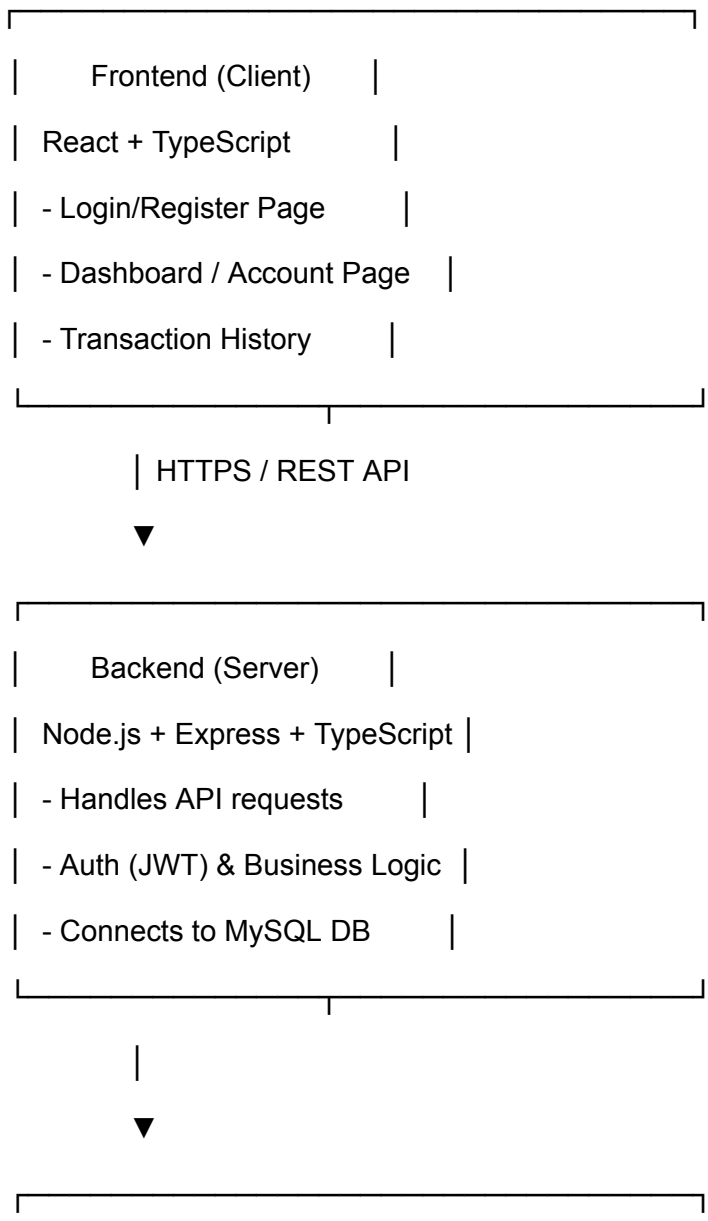
- **Security:** Use HTTPS, password hashing (bcrypt), and JWT tokens.
- **Performance:** Response time should be under 2 seconds for major operations.
- **Scalability:** Support multiple concurrent users efficiently.
- **Reliability:** Database must ensure data integrity during transactions.
- **Maintainability:** Code should be modular and containerized using Docker.
- **Portability:** The app should run on any environment supporting Docker.

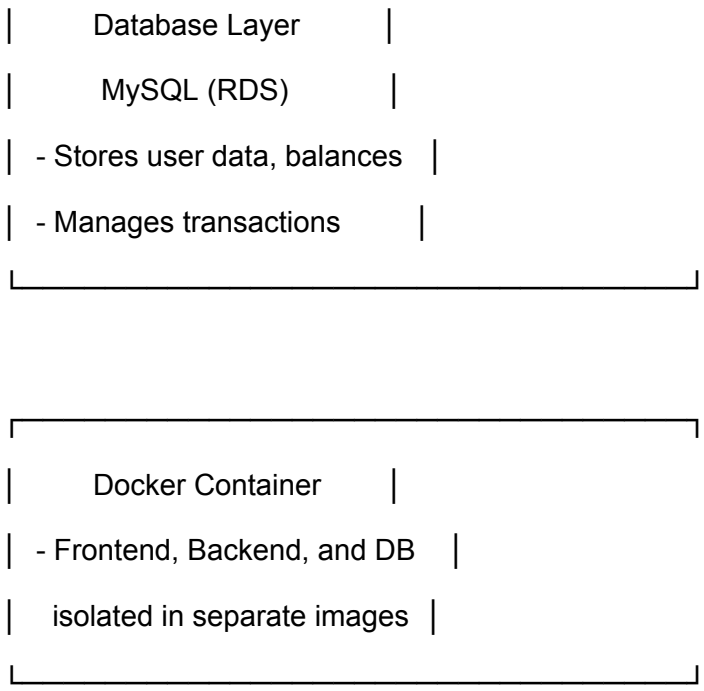
4. Key Architectural Decisions

Decision Area	Choice	Rationale
Frontend Framework	React + TypeScript	Provides component-based architecture and type safety.
Backend Framework	Node.js + Express	Asynchronous and scalable server-side development.
Database	MySQL	Relational structure ideal for managing banking transactions and relationships between users and accounts.

API Communication	RESTful APIs	Simplifies communication between frontend and backend.
Containerization	Docker	Ensures consistent deployment environment and scalability.
Authentication	JWT (JSON Web Tokens)	Provides stateless, secure authentication between client and server.

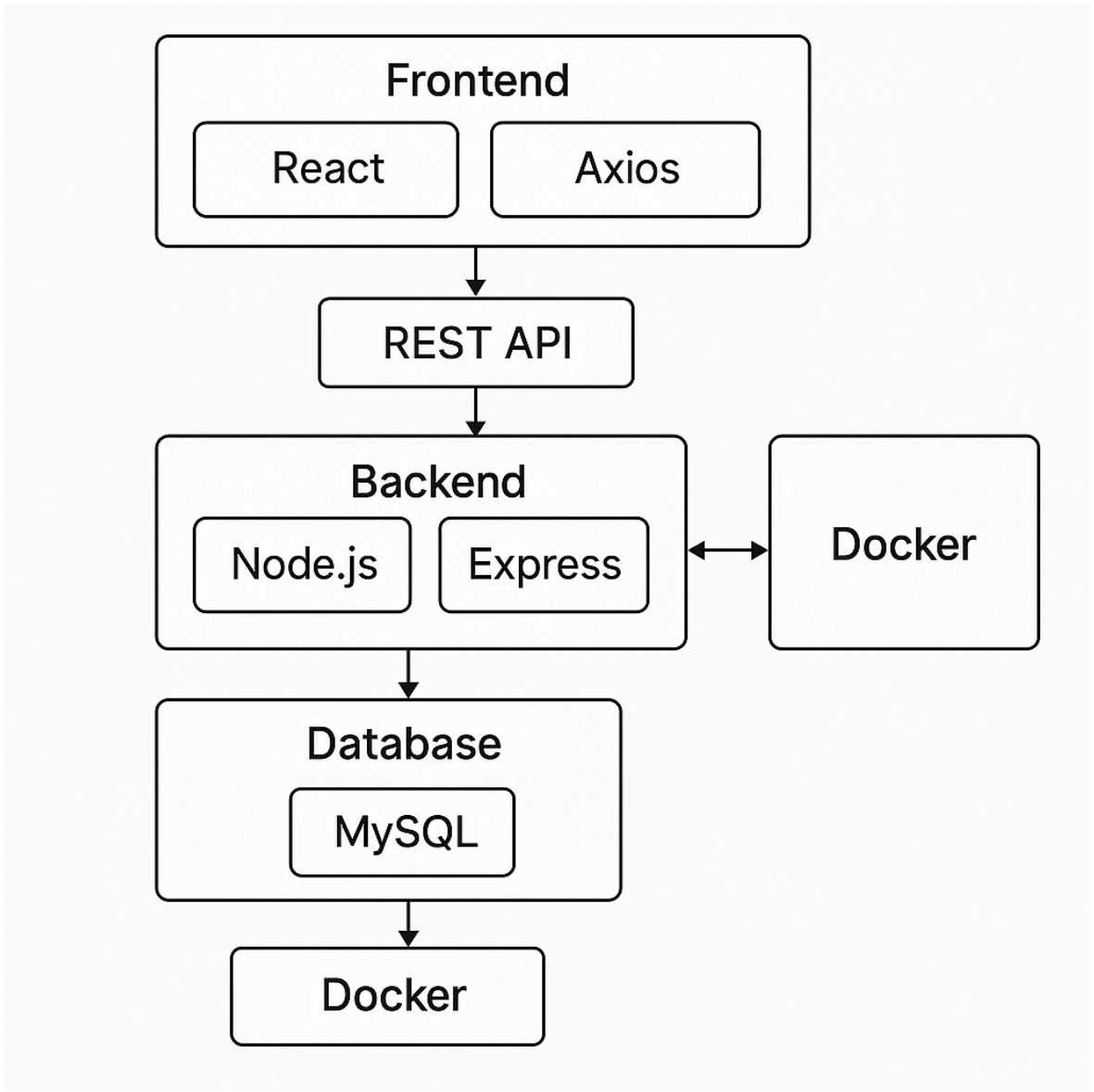
5. System Architecture Diagram





6. Tech Stack Summary

Layer	Technology	Purpose
Frontend	React, TypeScript	User interface, client-side rendering
Backend	Node.js, Express	API, server-side logic
Database	MySQL	Persistent data storage
Authentication	JWT	Secure login and access control
Containerization	Docker	Easy deployment and scaling


Frontend
React
Axios
REST API
Backend
Node.js
Express
Docker
Database
MySQL
Docker