**BANKING SYSTEM BACKEND DEVELOPMENT**

**Project Overview**

This Java Spring Boot-based system is a microservices architecture for managing client data, bank accounts, loans, transactions, and notifications. The API Gateway routes requests to various services like LoginService, ClientService, AccountService, TransactionService, and NotificationService, each handling specific tasks such as user authentication, client and account management, transaction processing, and client notifications. The system uses FeignClient for inter-service communication and likely persists data in a relational database, making it ideal for a financial application.

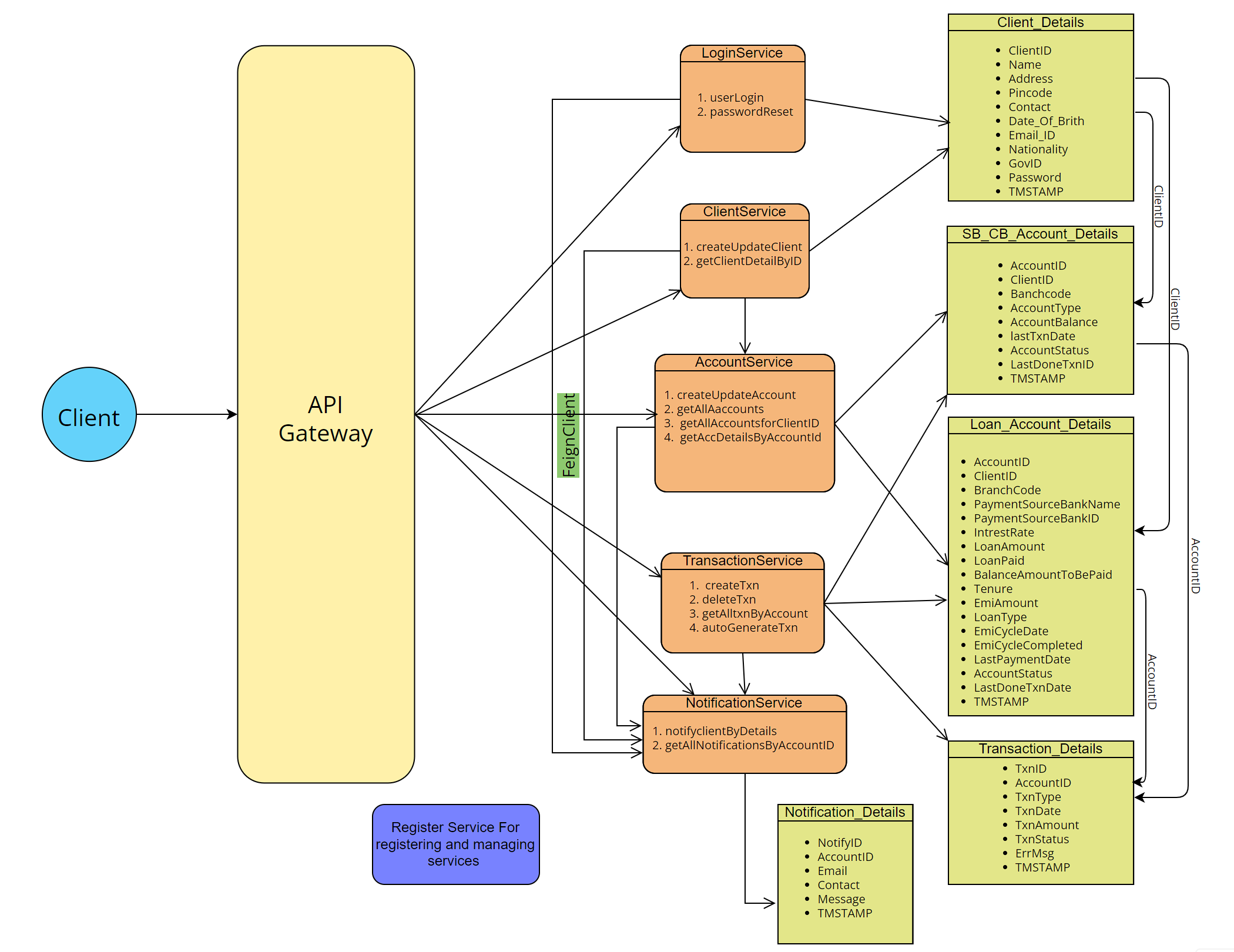
**Developer Details**

* designed & developed by Shubham Soman Samanta
* Email ID: [shubhamsamanta9302@gmail.com](mailto:shubhamsamanta9302@gmail.com)
* Contact: +91 8108442967
* Github:

**Contents:**

* **Phase 1**: Initial Design Overview

**Phase 1**: Initial Design Overview

The diagram outlines a Java Spring Boot-based microservices architecture for managing client data, bank accounts, loans, transactions, and notifications within a financial application.

**1.1 API Gateway**

* Acts as the entry point, routing client requests to the appropriate microservices and handling tasks like authentication and load balancing.

**1.2 Microservices**

* LoginService:
  + Manages user authentication, including login and password reset.
* ClientService:
  + Handles client data operations, such as creating, updating, and retrieving client details.
* AccountService:
  + Manages bank and loan accounts, including account creation, updates, and retrievals.
* TransactionService:
  + Oversees transaction processing, including creation, deletion, and retrieval of transactions.
* NotificationService:
  + Manages client notifications, sending alerts and retrieving notification history.

**1.3. Data Model**

* Client\_Details:
  + Stores client information.
* SB\_CB\_Account\_Details:
  + Manages savings and current bank account details.
* Loan\_Account\_Details:
  + Tracks loan-related account information, including payments and balances.
* Transaction\_Details:
  + Logs transactions linked to accounts.
* Notification\_Details:
  + Captures notifications sent to clients.

**1.4. Inter-service Communication**

* Services communicate via REST APIs, with FeignClient used for HTTP requests between services.

**1.5. Database and Persistence**

* The system likely uses a relational database, with each microservice managing its data model entities.

This architecture provides a modular, scalable solution for managing clients and their financial interactions, making it ideal for financial institutions.