**BANKING-APPLICATI0N**

This document describes the complete **microservices architecture** of the Banking Application, including Discovery and Config servers, API Gateway, Auth, Customer, Account, Payment, Audit, Feedback, and Notification services. It also explains Angular frontend integration, Kafka messaging, and deployment architecture.

**GlobalSystemOverview:**The Banking Application is implemented as a microservices ecosystem: - Angular Frontend for user interface - API Gateway as a single-entry point - Discovery Server for service registry - Config Server for centralized configurations - Domain-driven microservices (Customer, Account, Payment, etc.) - Kafka and Zookeeper for messaging.

**Discovery &ConfigServices-** Discovery Service (Eureka): Registers and discovers microservices dynamically. - Simplifies API Gateway routing by avoiding hard-coded URLs. - Enables load balancing. Config Server: Centralized configuration management for all microservices. - Externalized properties for different environments (dev, QA, prod). - Integrated with Git for version control.

**API Gateway**- Acts as a single entry point for client (Angular) requests. - Performs authentication, routing, and load balancing. - Handles CORS, rate limiting, and logging.

**Auth Service-** Provides login, registration, and JWT token generation. - Integrates with API Gateway to enforce authorization policies. - Uses Spring Security for authentication.

**Customer Service-** Stores and manages customer profiles and KYC information. - Supports CRUD operations via REST endpoints. - Communicates with AuditService to log all profile updates.

**Account Service-** Manages account data including balances and account types. - Exposes APIs for fetching and updating balances. - Called by PaymentService for fund validation.

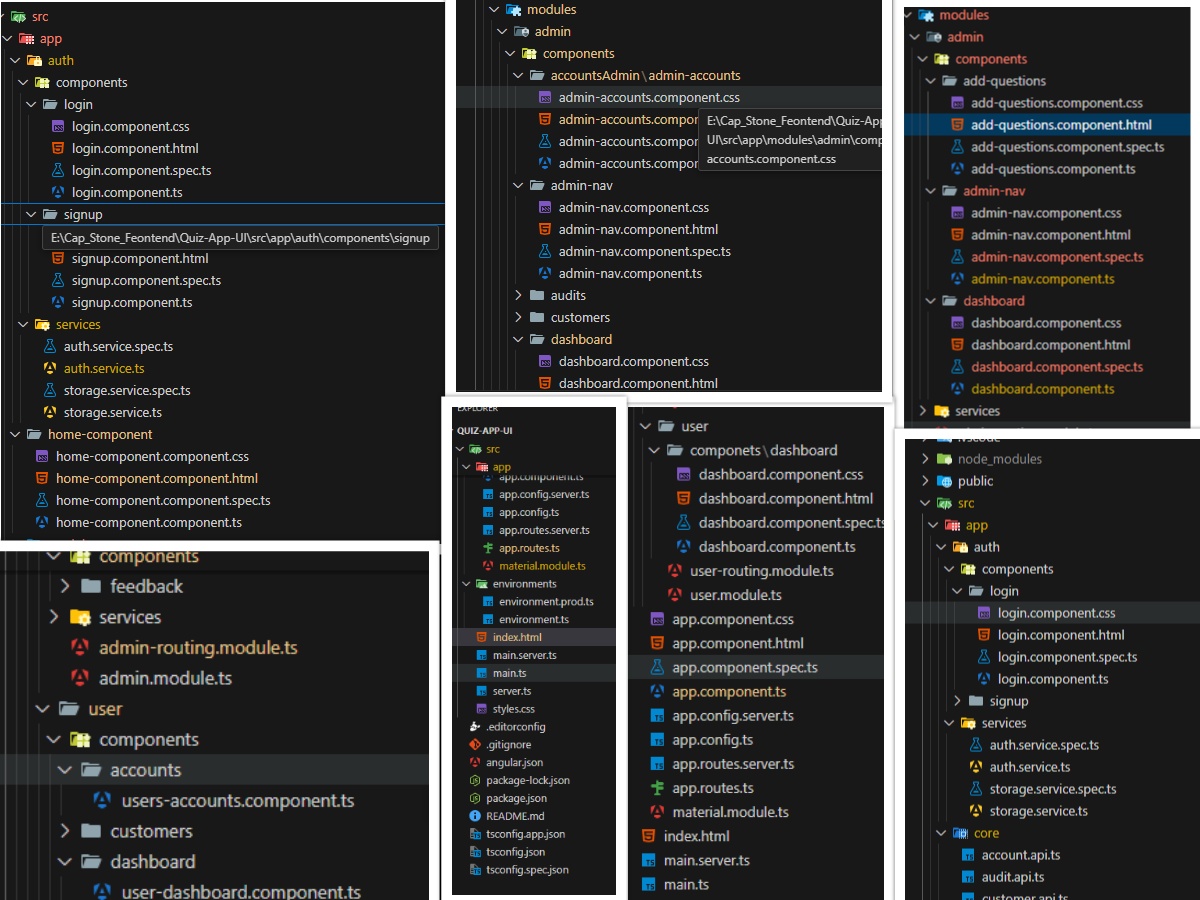
**Payment Service-** Handles fund transfers and payment workflows. - Communicates with AccountService, AuditService, and NotificationService. - Publishes Kafka events for asynchronous notifications.

**Audit Service**- Maintains logs for all critical actions and transactions. - Provides APIs for querying audit records. Integrated with PaymentService and CustomerService. Feedback Service- Collects and stores customer feedback. - Communicates with NotificationService to send alerts based on negative feedback. Notification Service- Sends notifications via email. - Listens to Kafka events (e.g., payment confirmation)

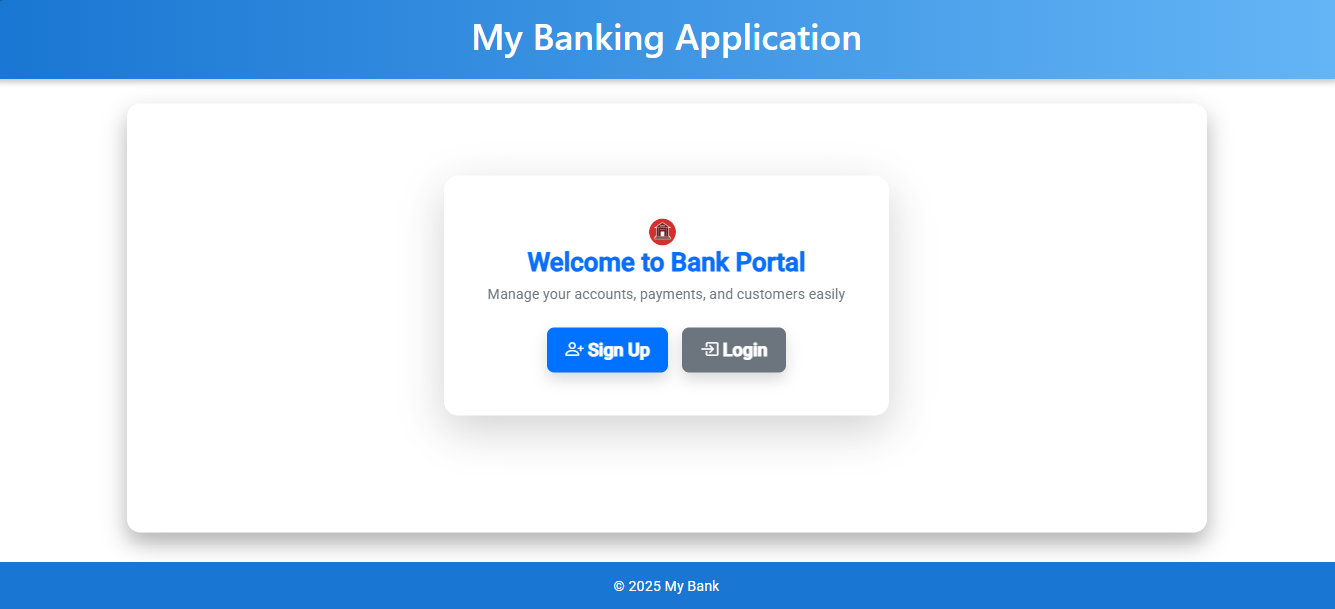
**Angular-Backend Communication Flow** 1. Angular sends authentication request via API Gateway. 2. JWT token is issued by AuthService and stored by Angular. 3. Angular makes secured requests to Gateway with token headers. 4. Gateway routes requests to respective services (Customer, Payment, etc.). 5. Kafka events notify NotificationService asynchronously.

**ANGULAR VIEW OUTPUTS**

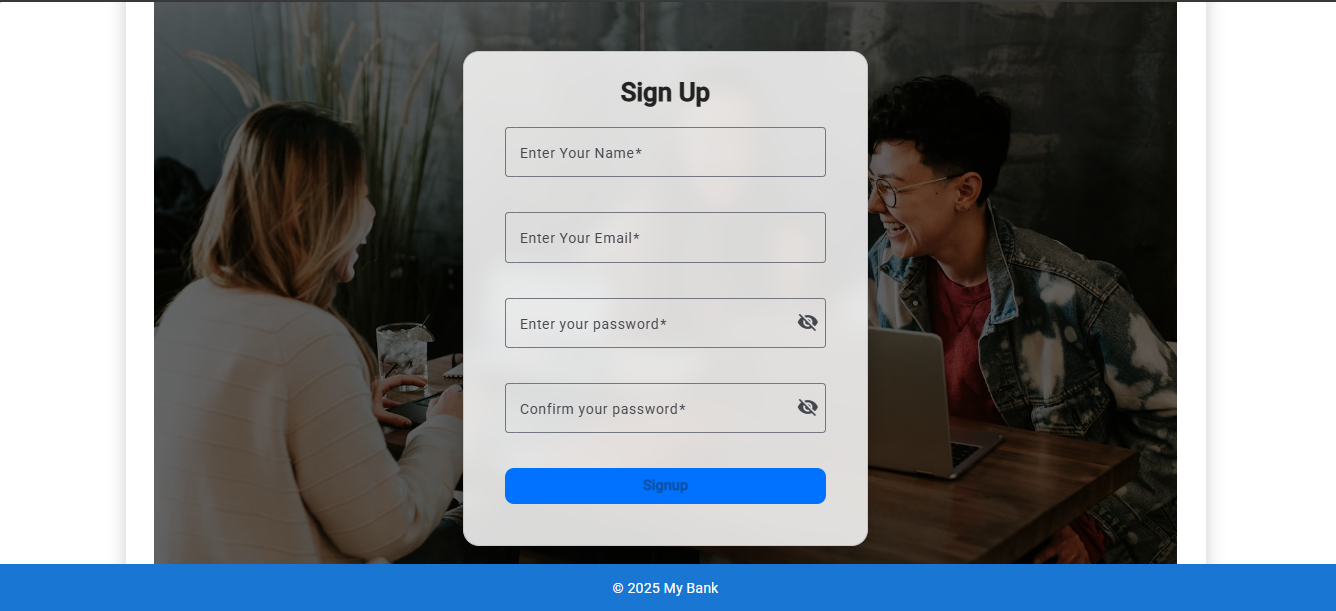
Folder Structure:

****

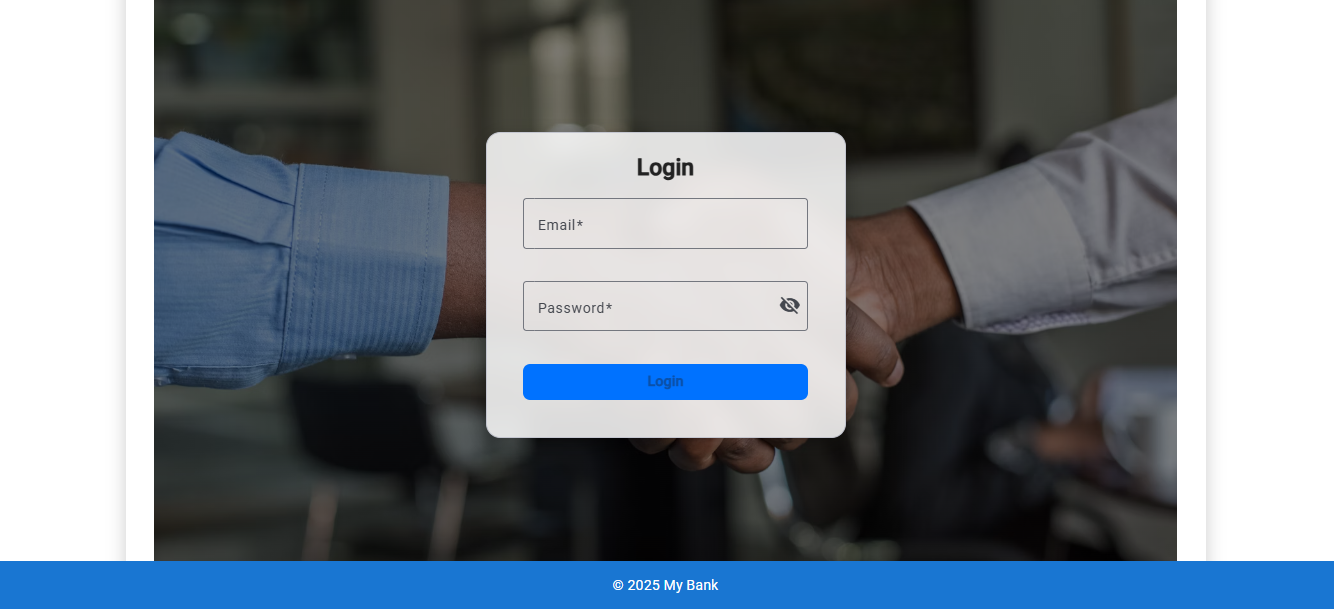
**HOME PAGE:**

****

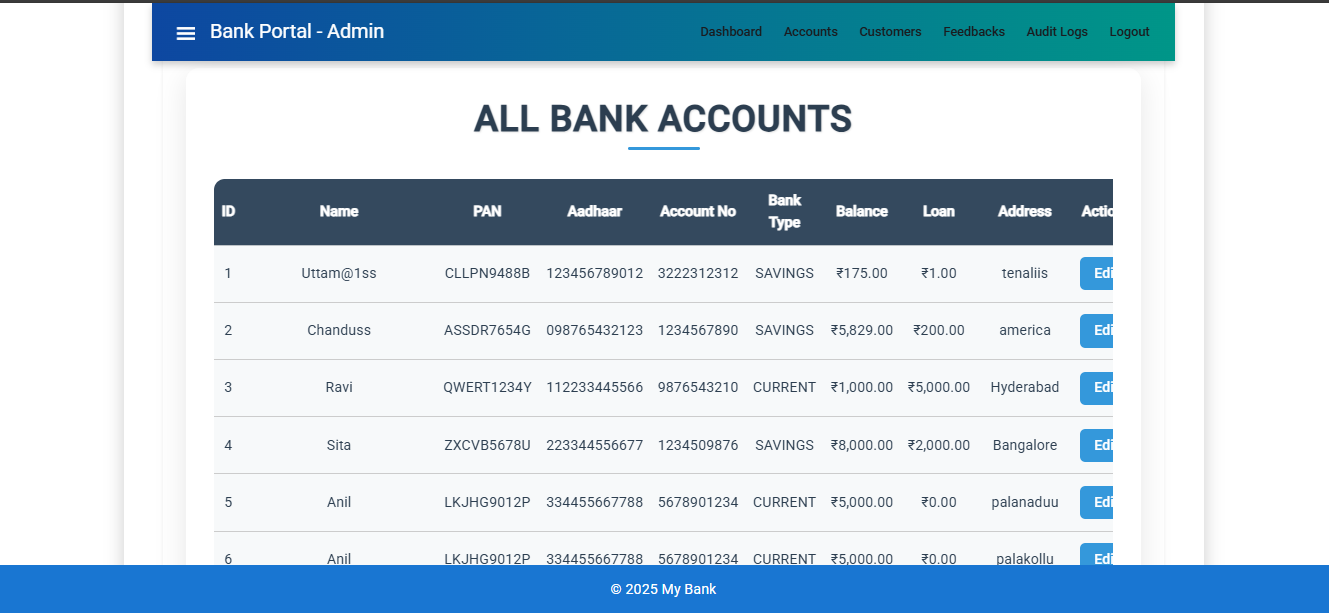
**SIGNUP PAGE:**

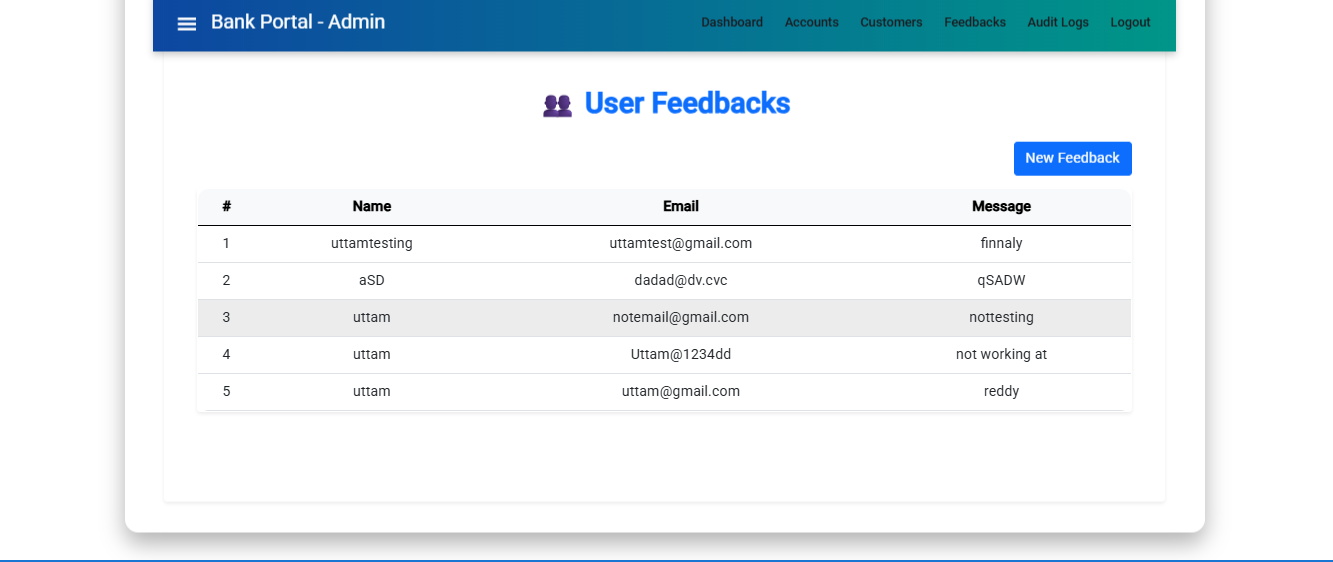
****

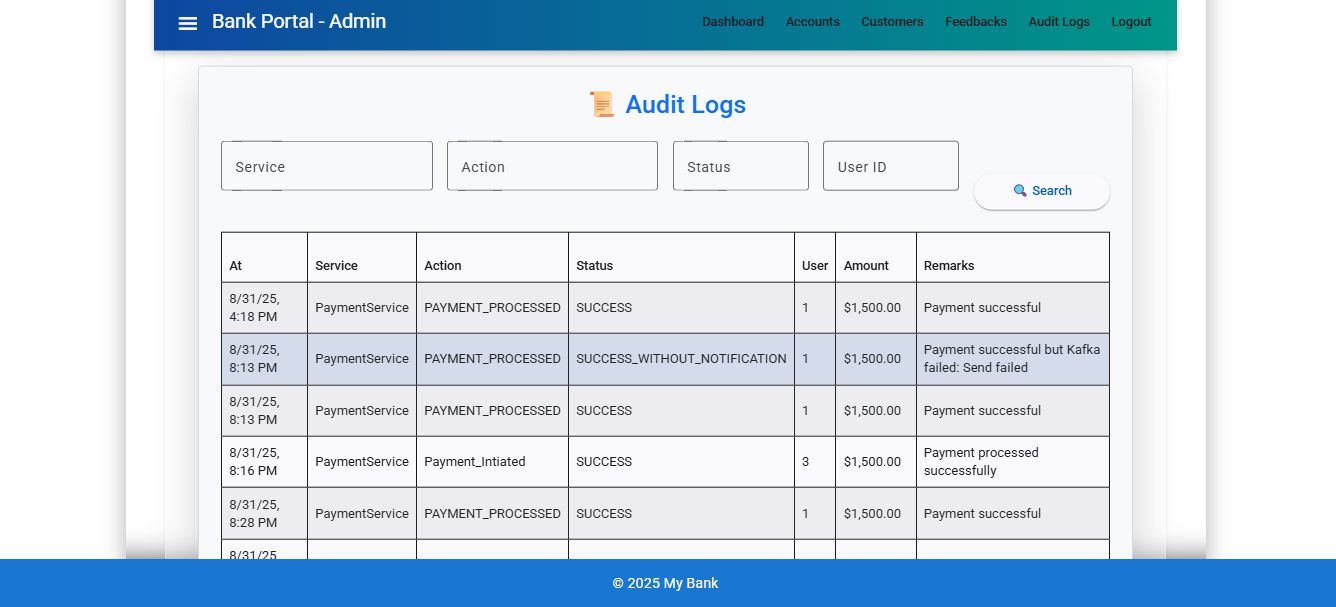
**LOGIN PAGE:**

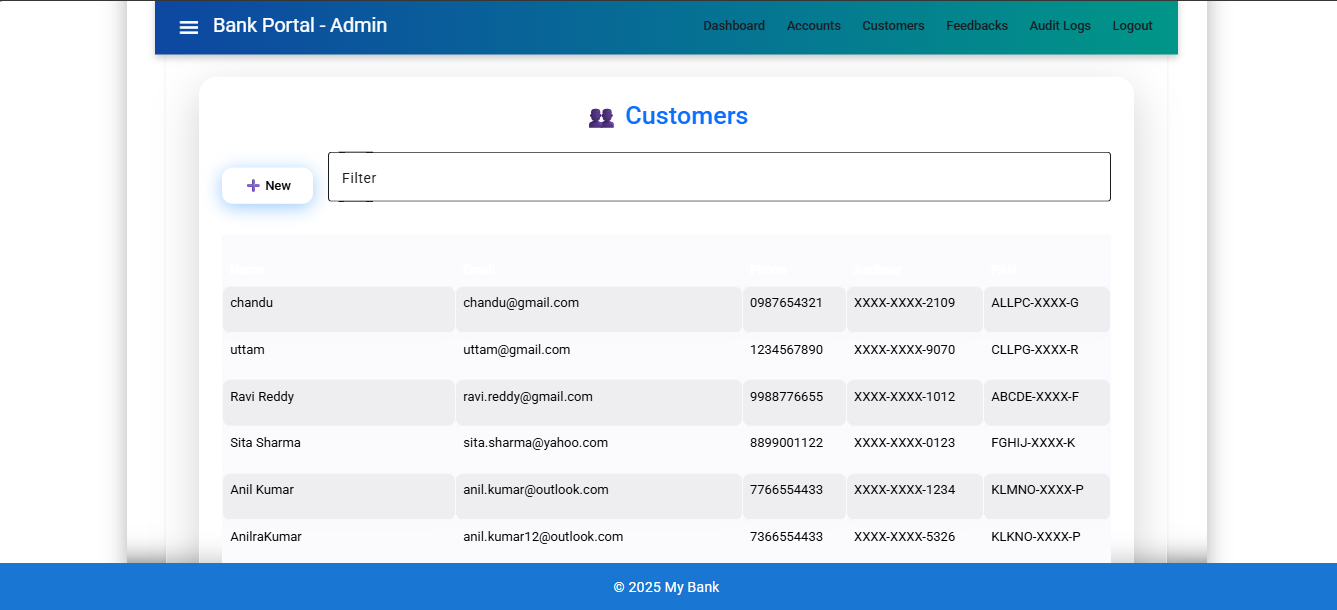
****

**ADMIN COMPONENTS VIEWS:**

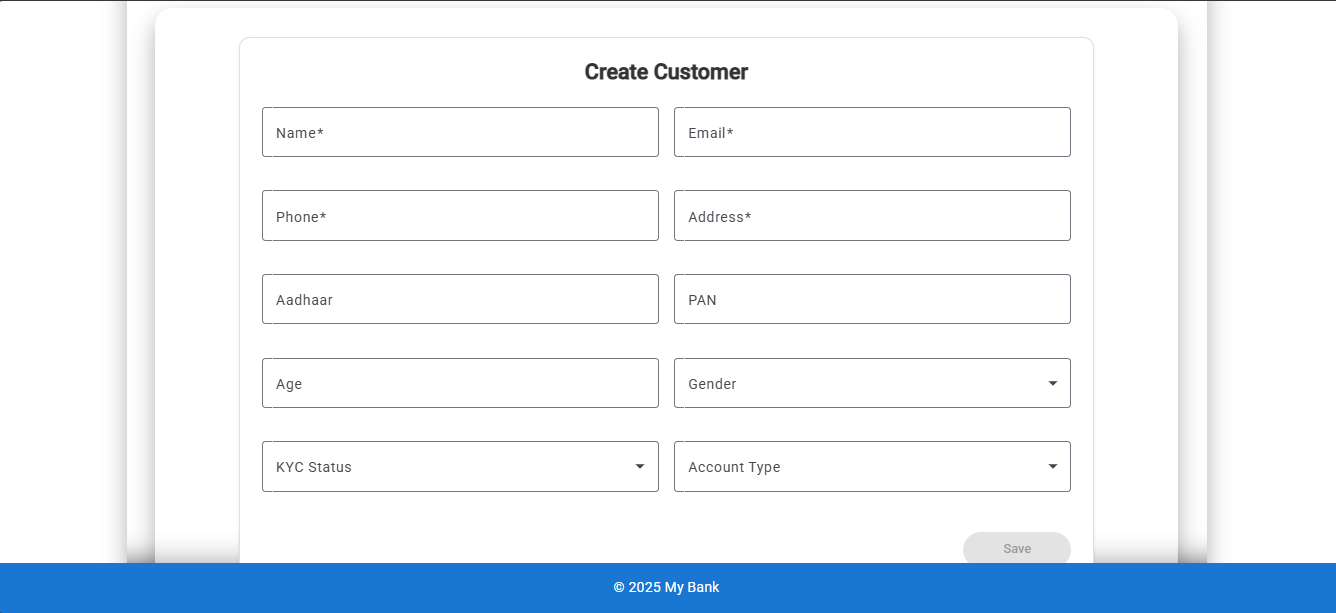
****

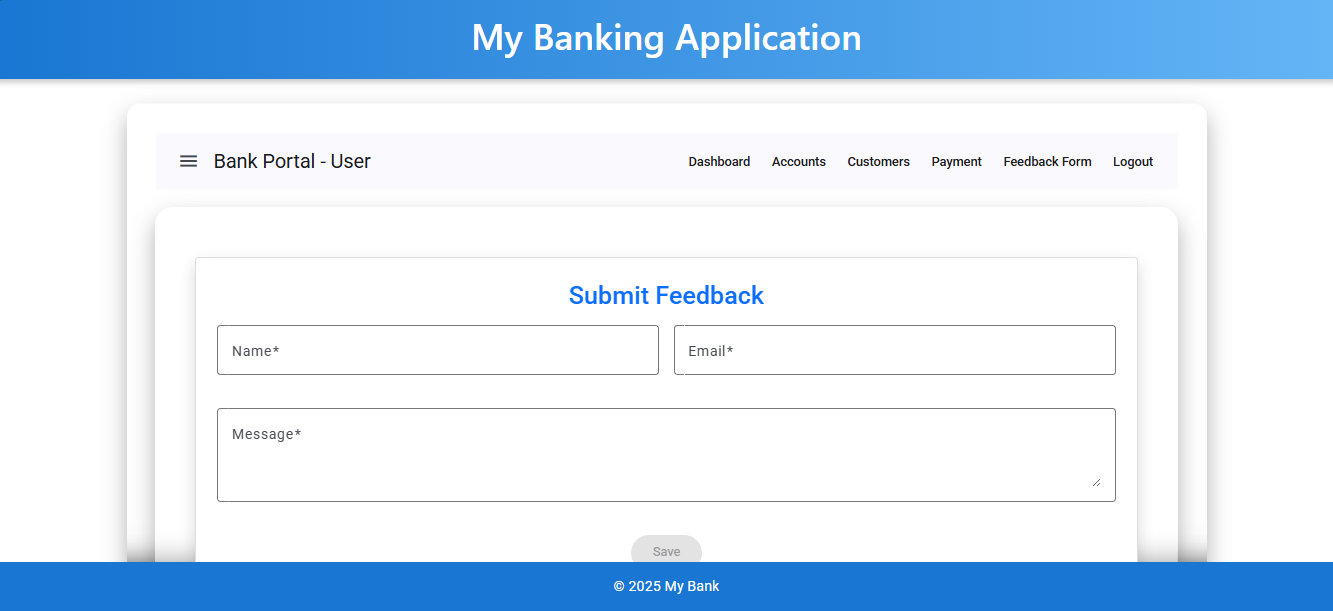
****

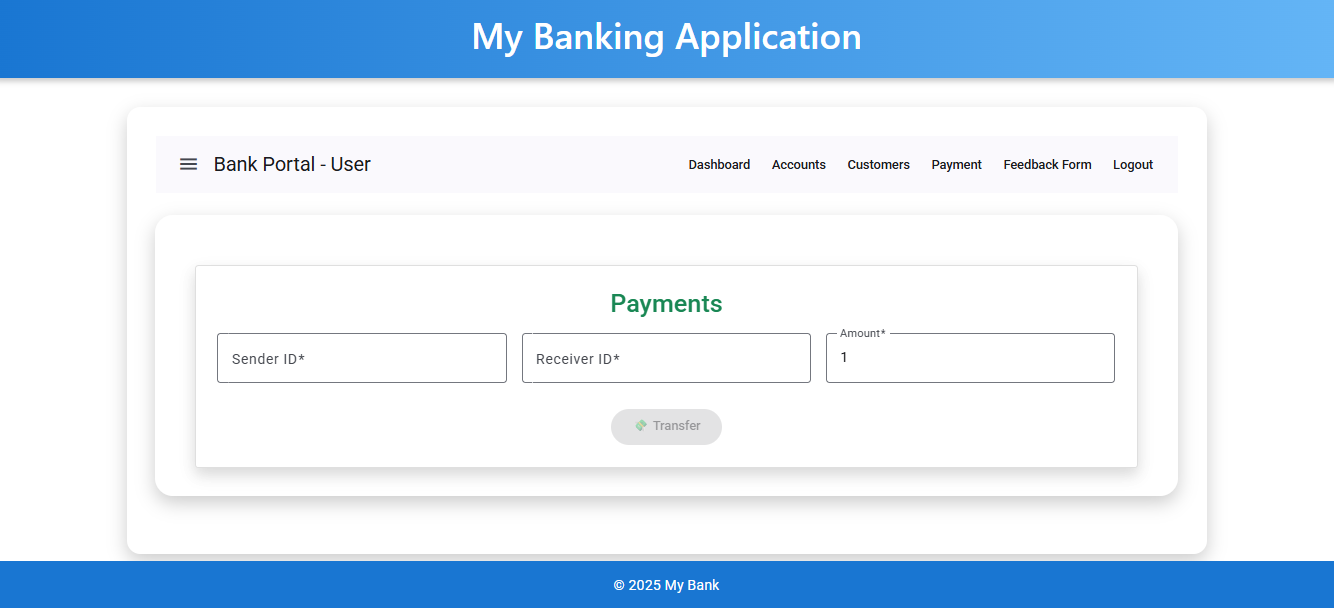
****

****

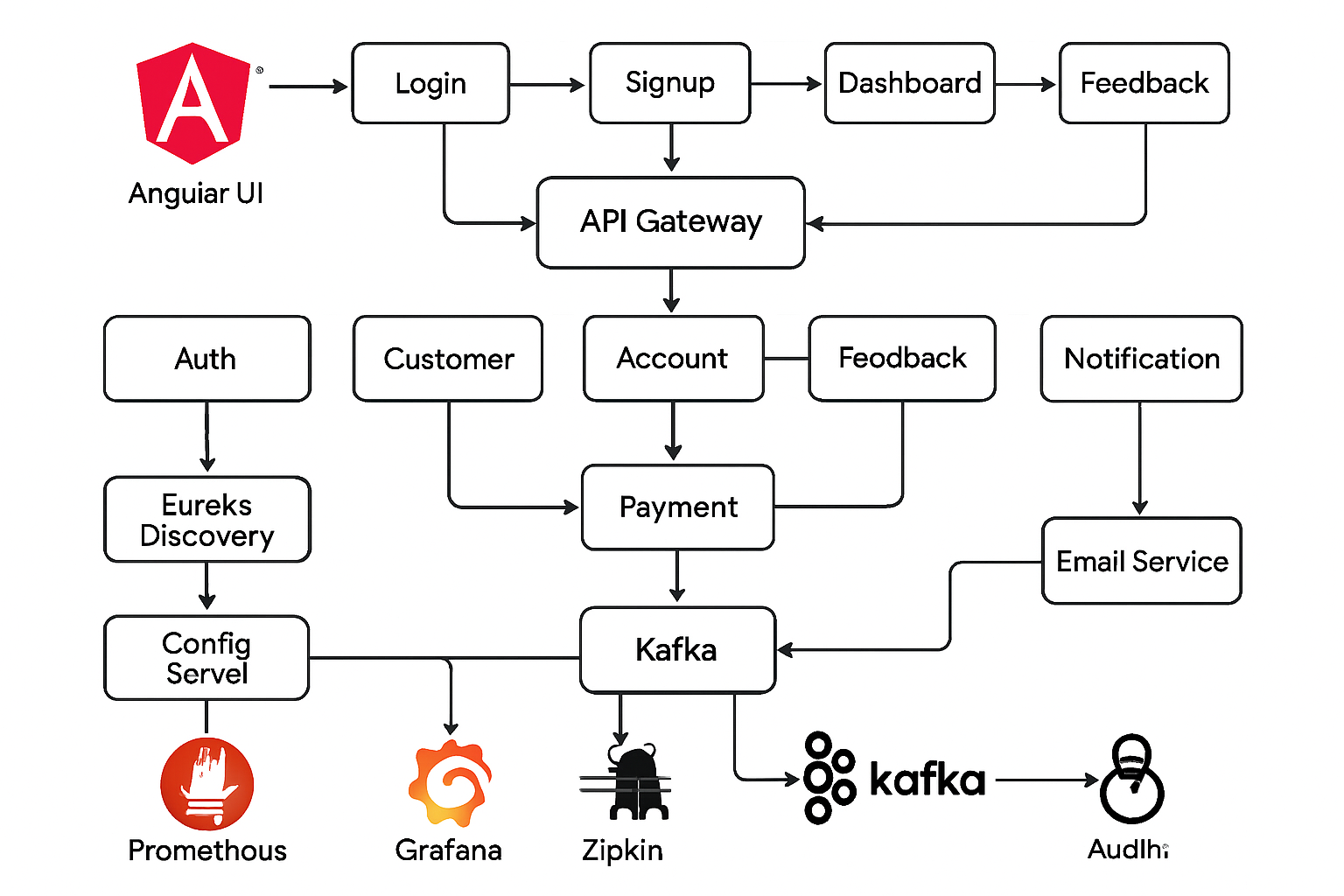
**USER COMPONENTS VIEWS:**

****

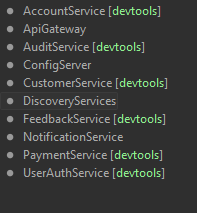
****

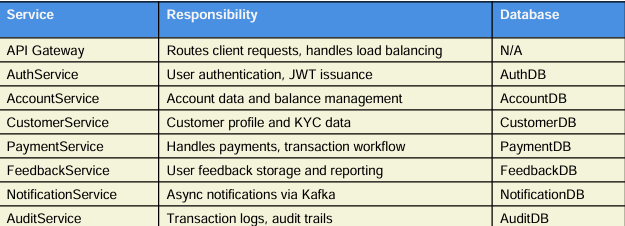
****

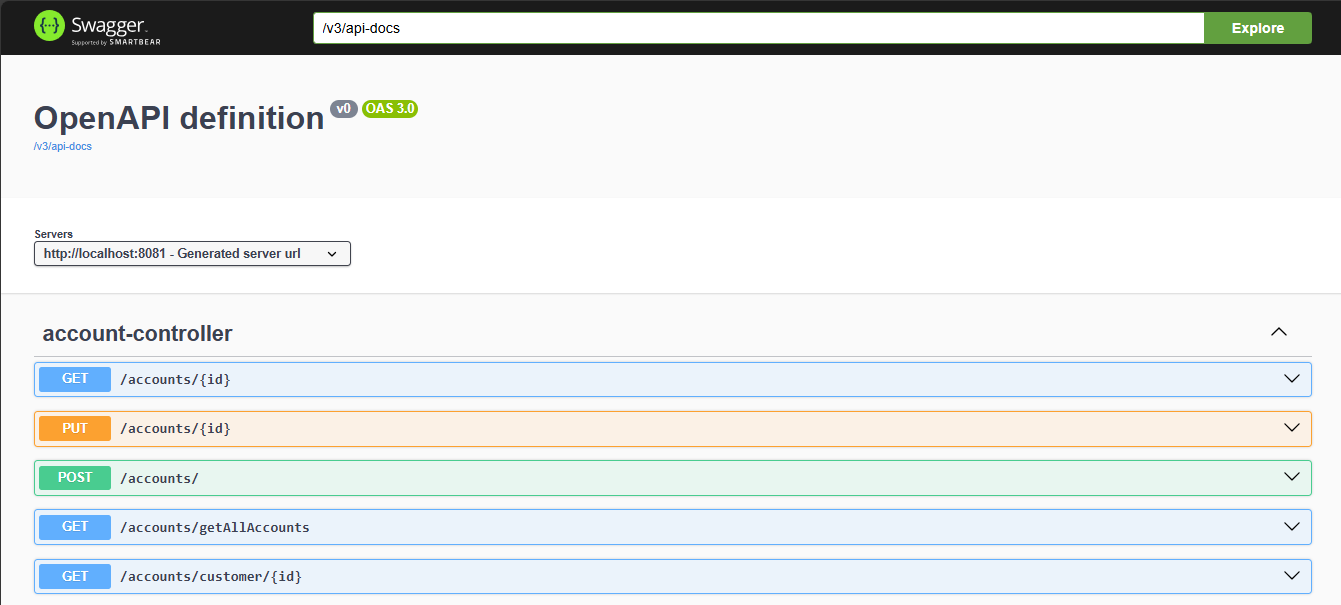
**ARCHITECTURE:**

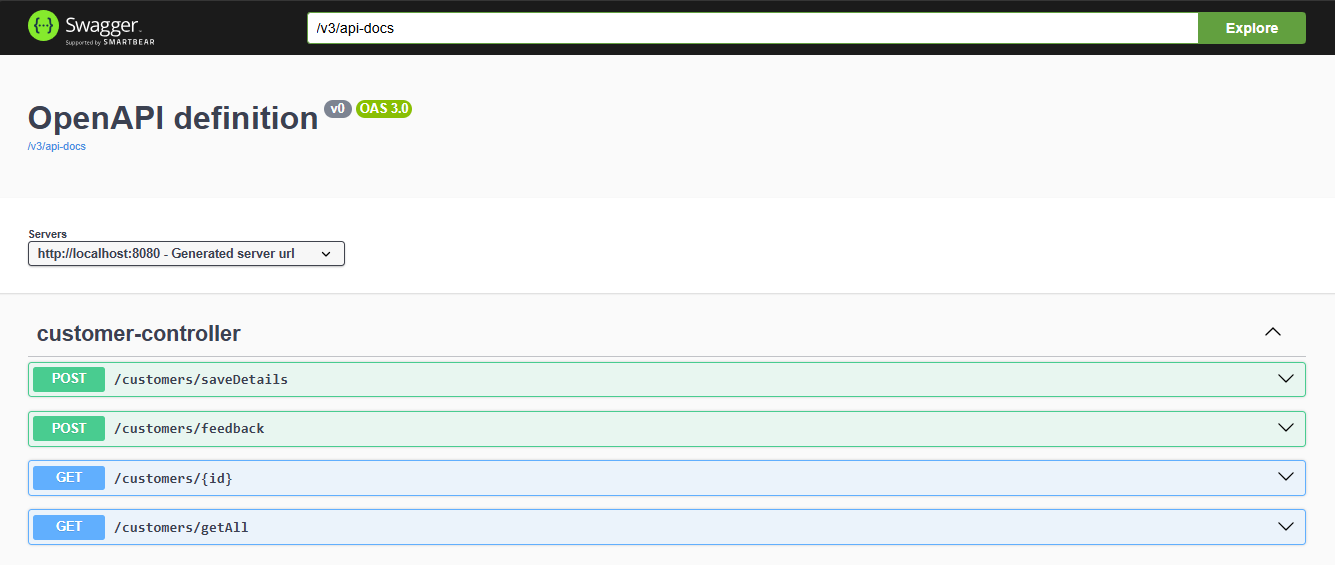
****

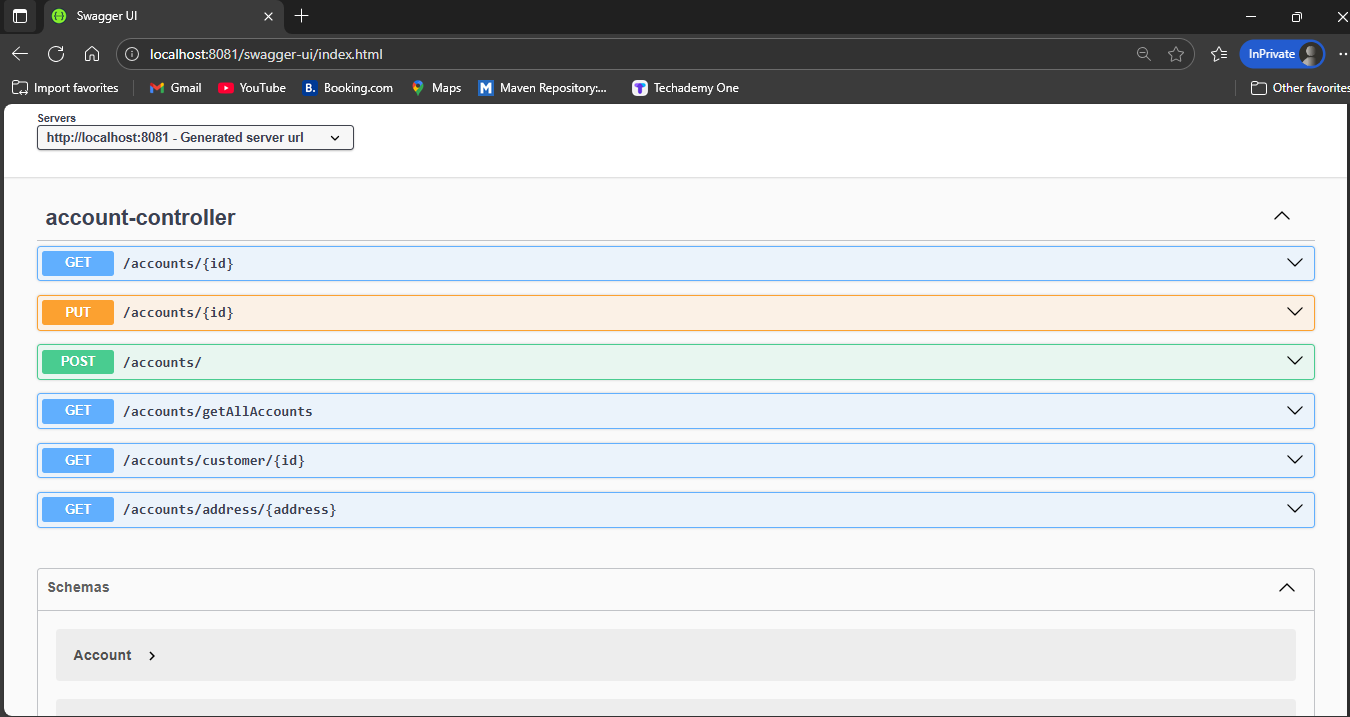
**JAVA-SPRING\_BOOT\_MICROSERVICES:**

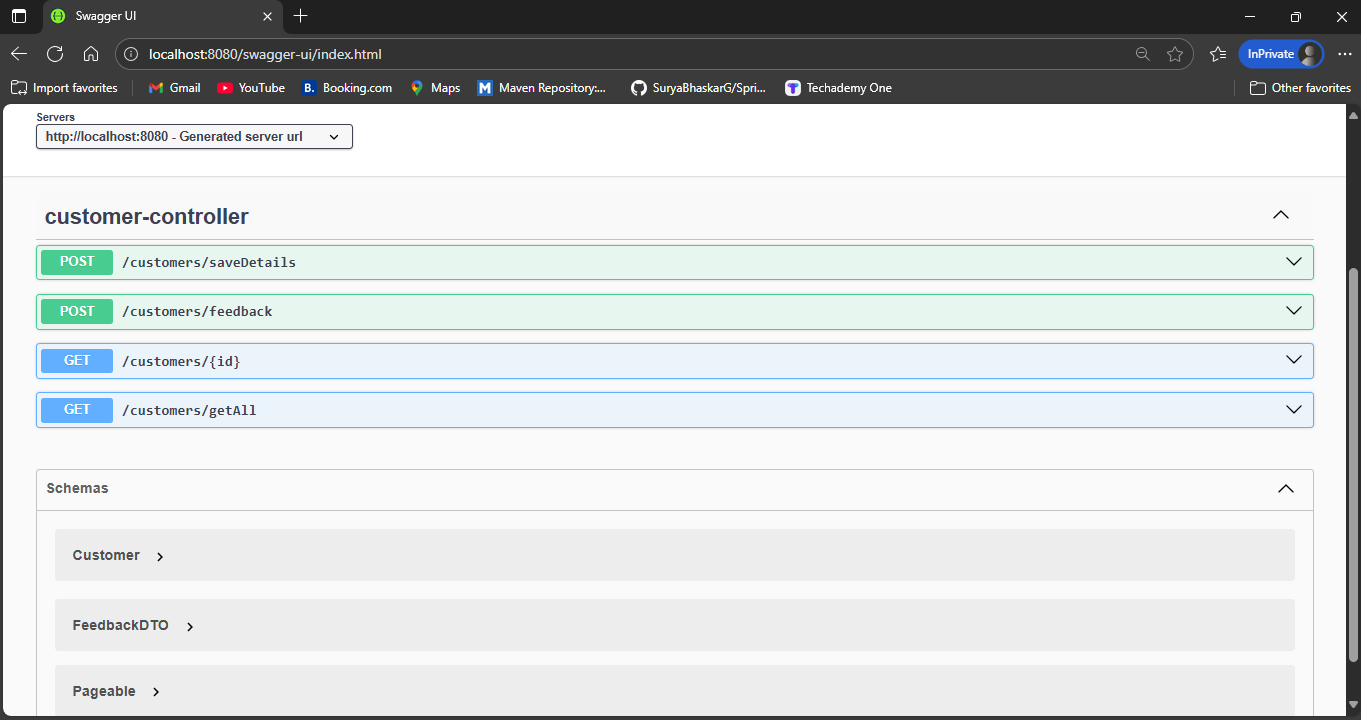
**SERVICES:**

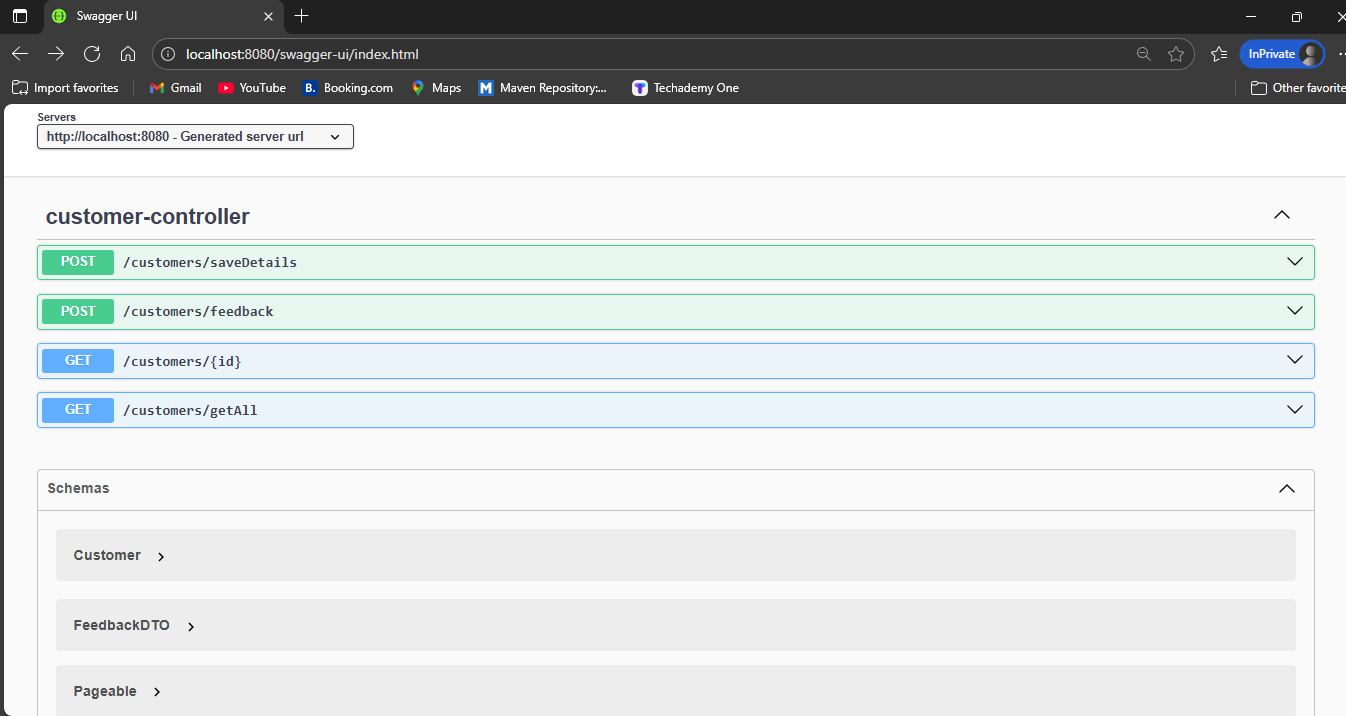
****

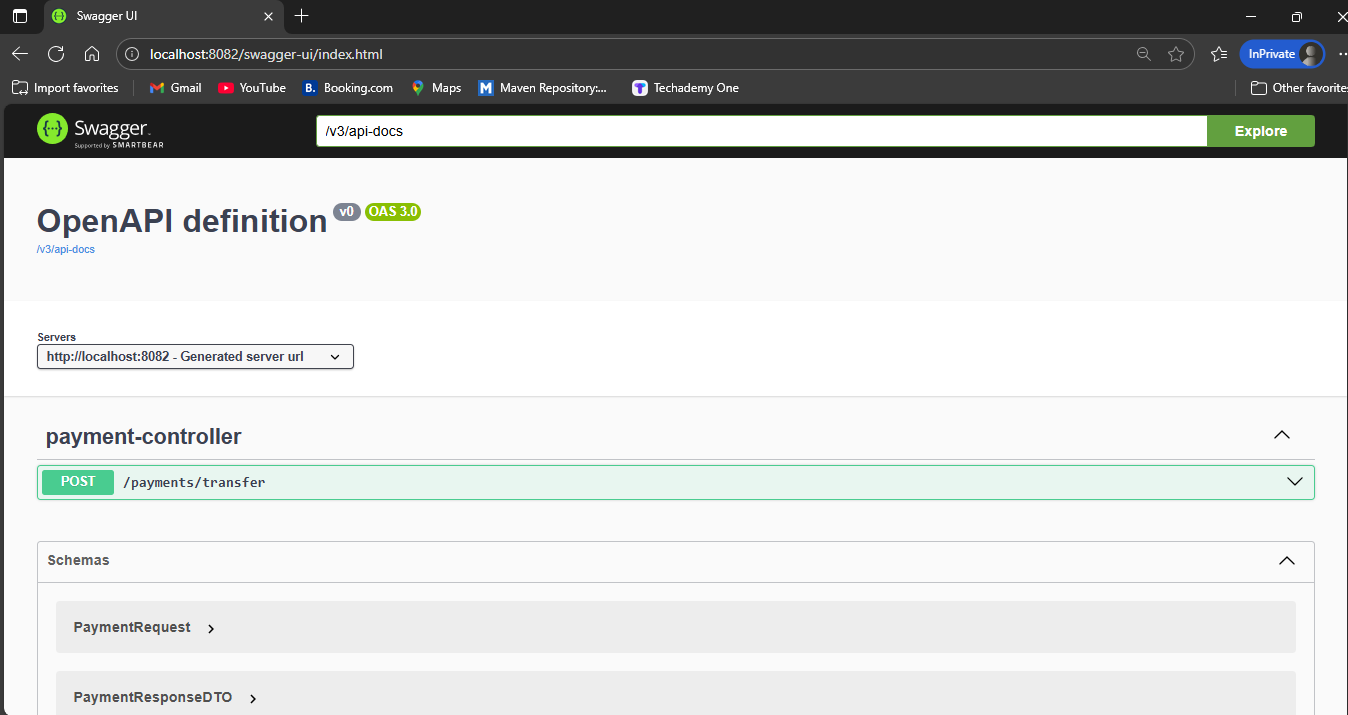
**SWAGGER and ZIPKIN and PROMETHEUS and GRAFANA :**

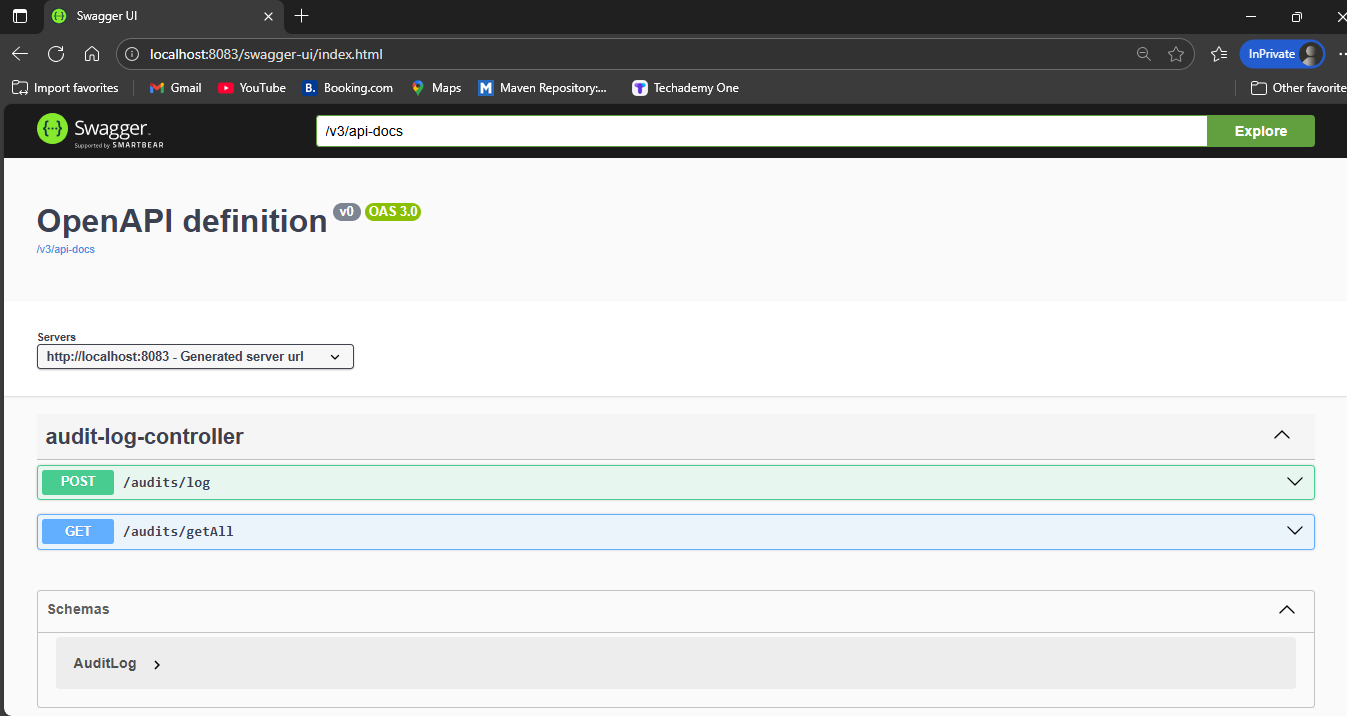
****

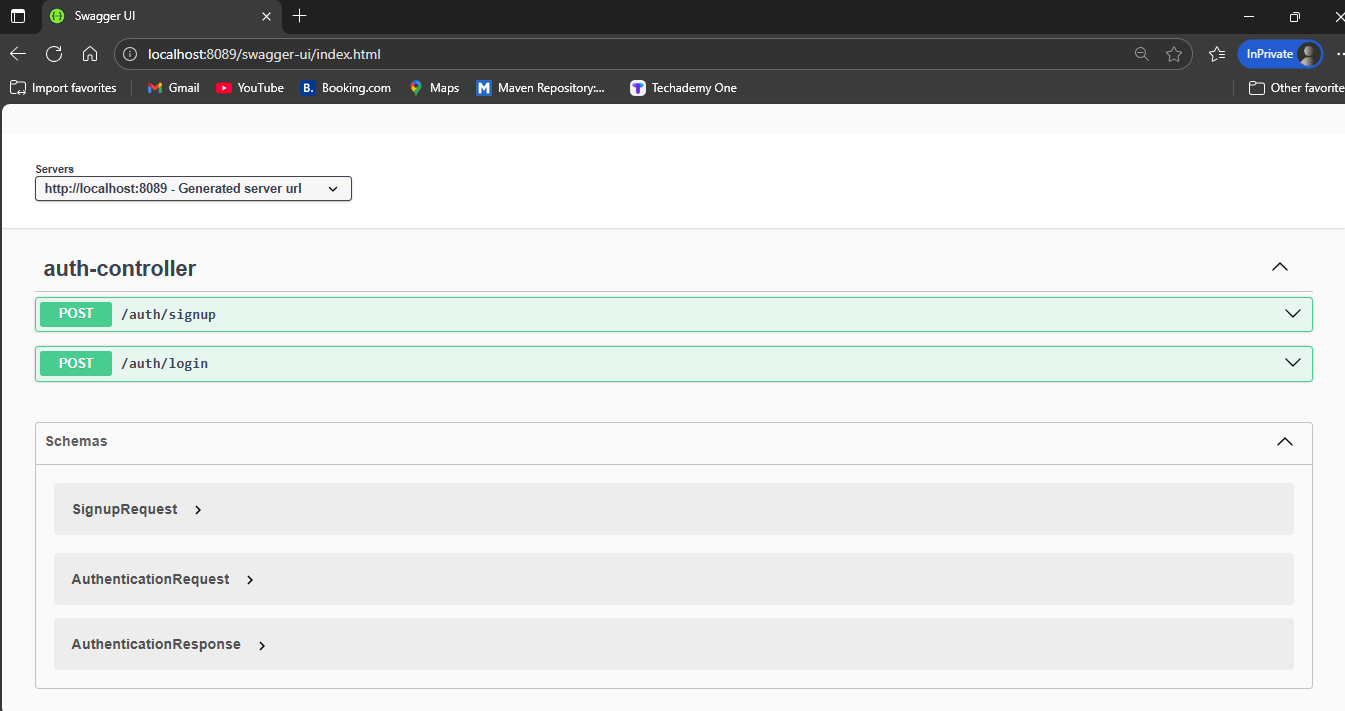
****

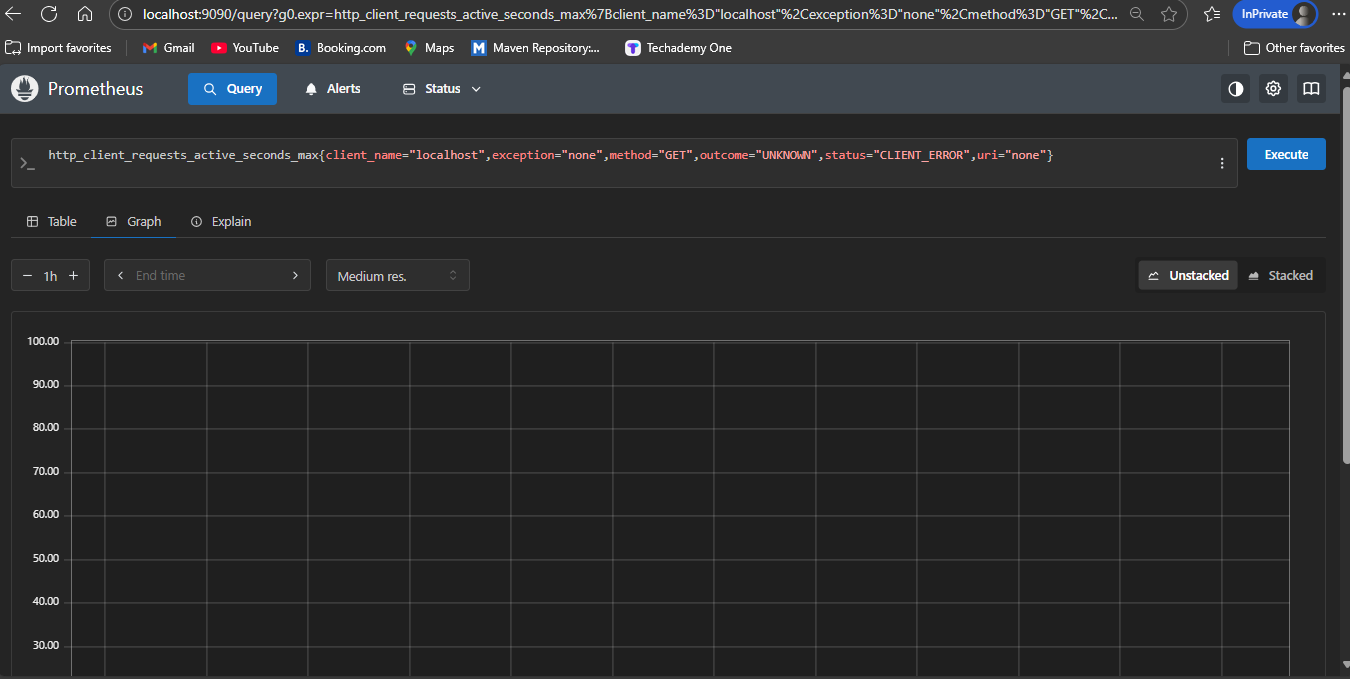
****

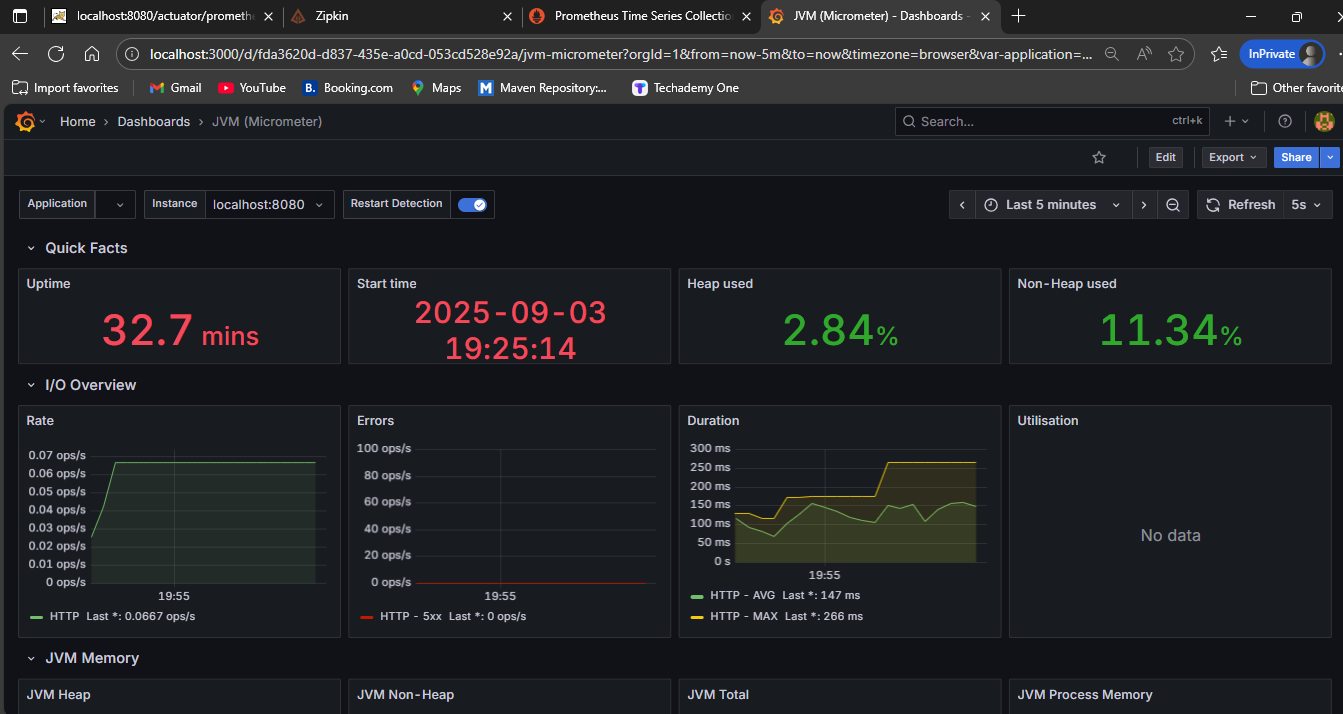
****

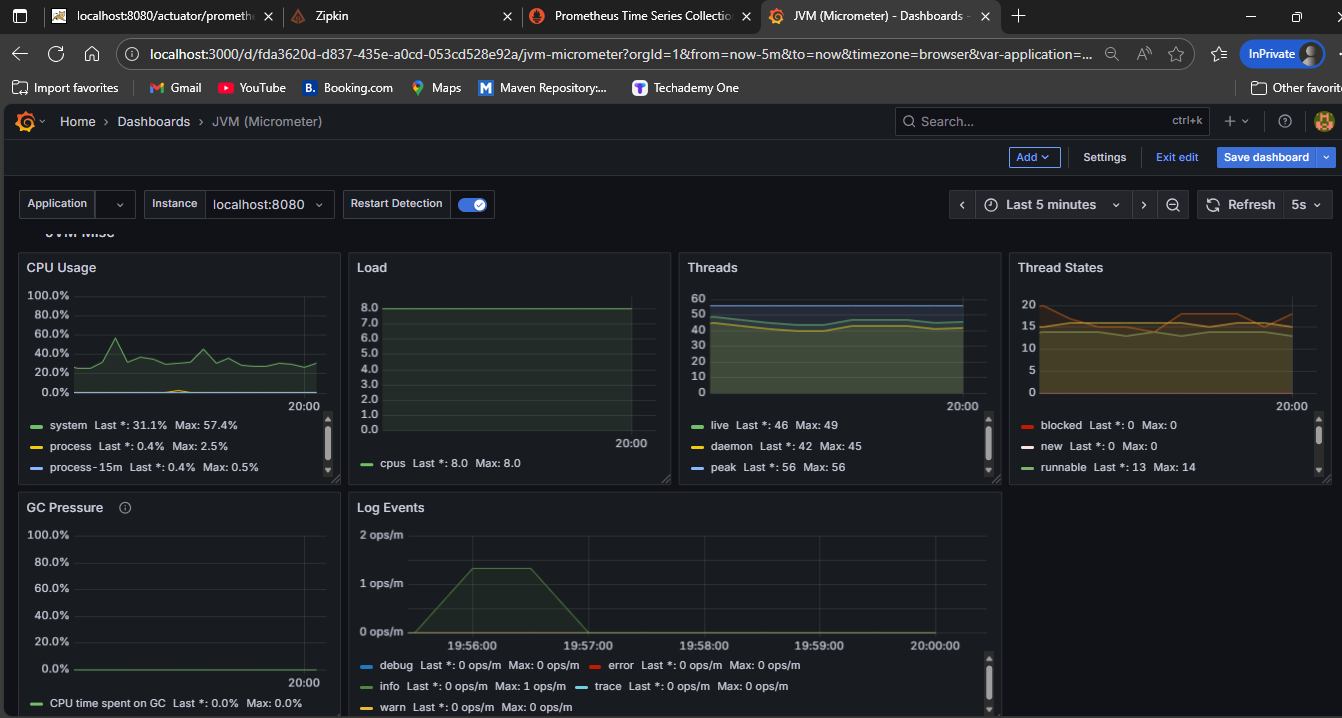
****

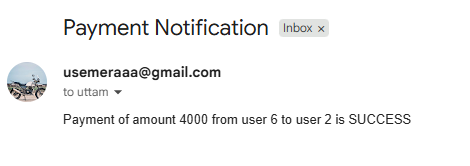
****

****

****

****

****

****



**OVERVIEW:**

The Banking Web Application project demonstrates a robust, microservices-based architecture built with Angular and Spring Boot. Core functionalities, including customer management, accounts, payments, feedback, audit, and notifications, are seamlessly integrated, with Kafka enabling asynchronous communication between Payment and Notification services and Feign Clients facilitating synchronous inter-service calls. Supported by API Gateway, Config Server, and Discovery Service, the system is secure, scalable, and maintainable. This project highlights modern enterprise-level development practices and provides a solid foundation for real-world banking solutions.

**Acknowledgment / Conclusion:**

This project has been completed successfully under the guidance of **Ramakrishna Sir**. I have applied my knowledge of **Angular, Spring Boot, and Microservices** to develop a scalable and secure Banking Web Application. I hope this project meets the objectives and expectations outlined at the start.

**Submitted By:**  
**Uttam Kumar Narra BATCH \_II**[**uttamnarra1@gmail.com**](mailto:uttamnarra1@gmail.com) **ANGULAR**