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.

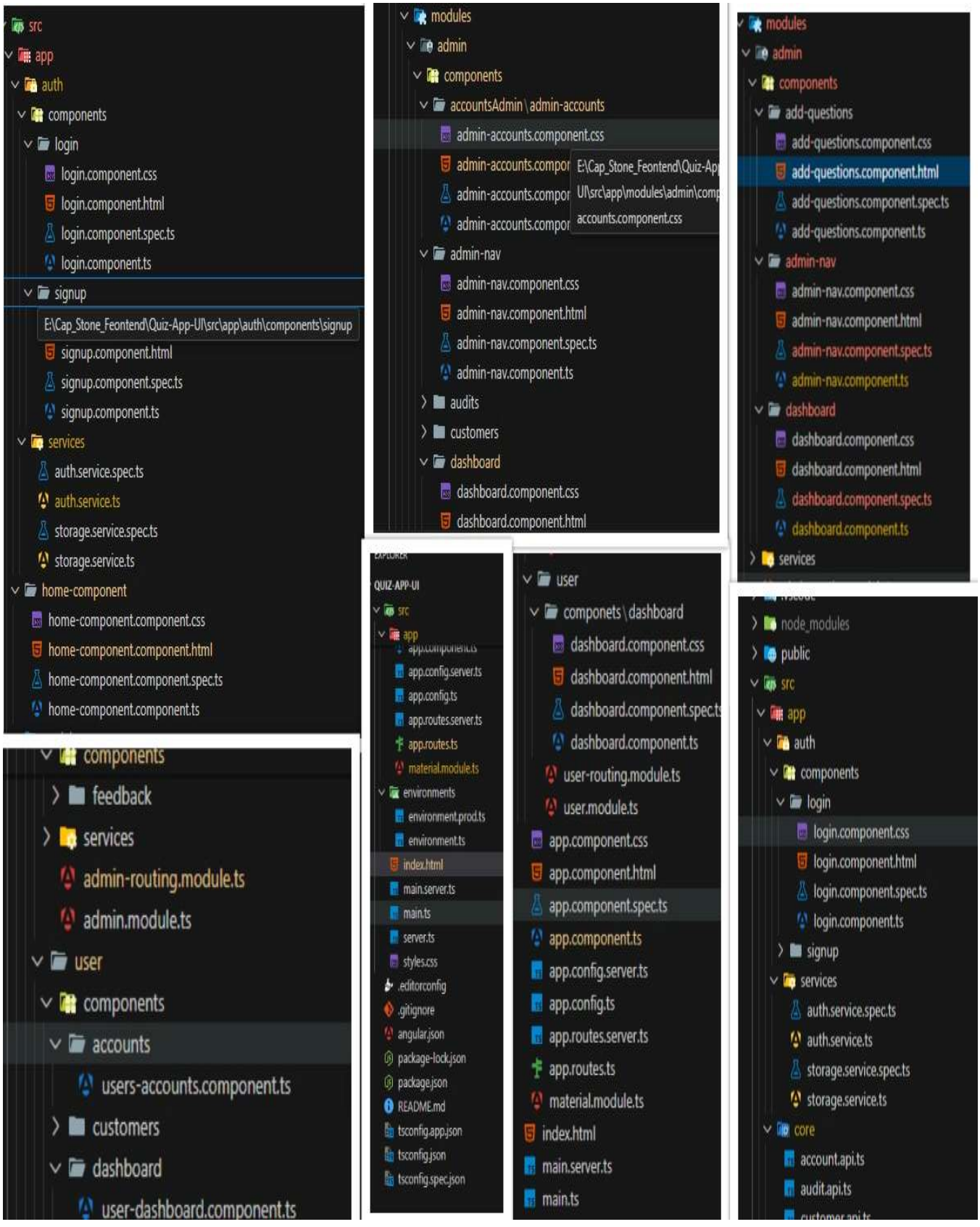
Payment Service- Handles fund transfers and payment workflows. - Communicates with AccountService, AuditService, and NotificationService. - Publishes Kafka events for asynchronous notifications.

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)

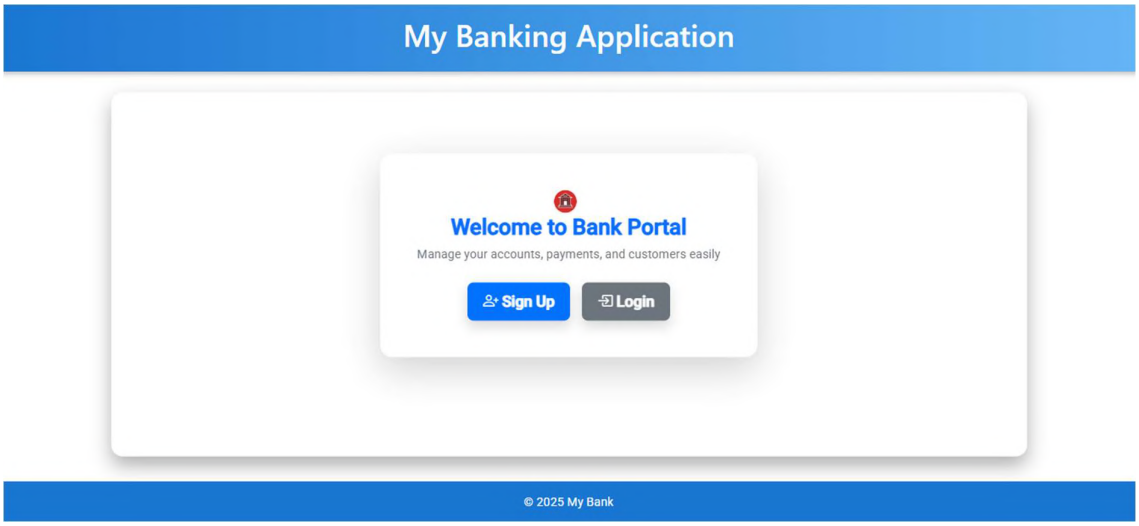
Angular-Backend Communication Flow 1. Angular sends authentication request via API Gateway. 2. JWT token is issued by AuthService and stored by Angular. 3. Angular makes secured requests to Gateway with token headers. 4. Gateway routes requests to respective services (Customer, Payment, etc.). 5. Kafka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

ANGULAR VIEW OUTPUTS

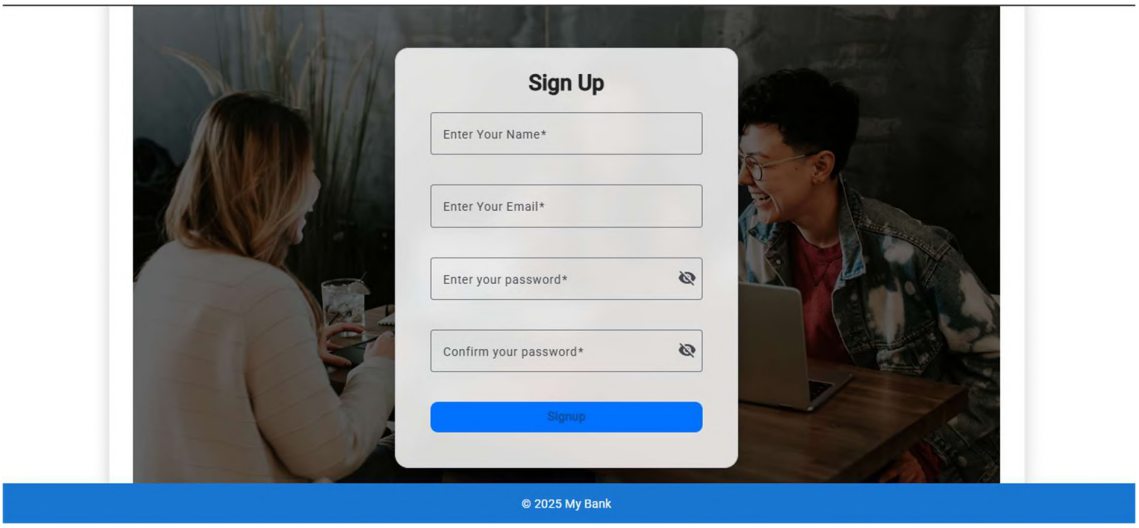
Folder Structure:



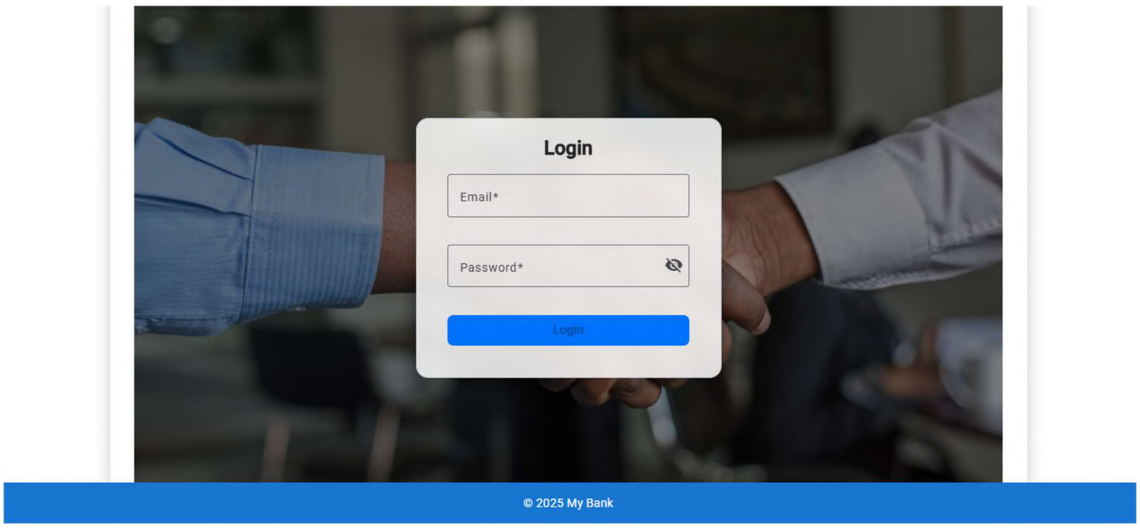
HOME PAGE:



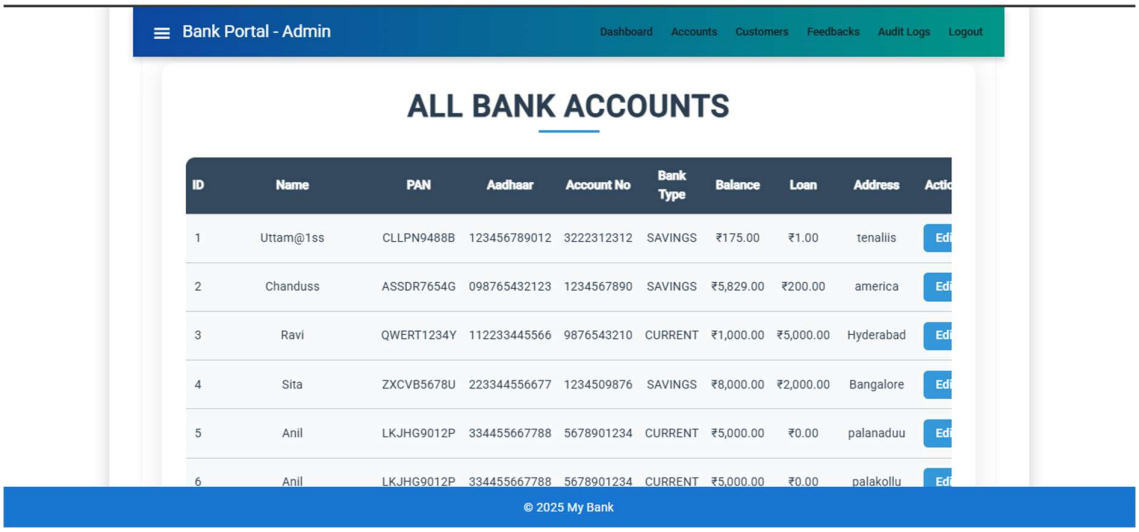
SIGNUP PAGE:

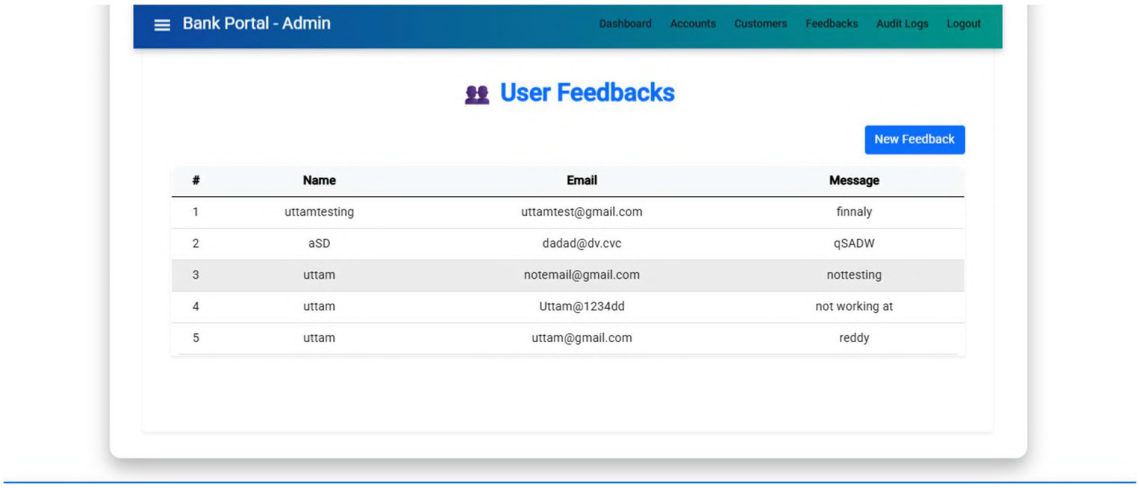


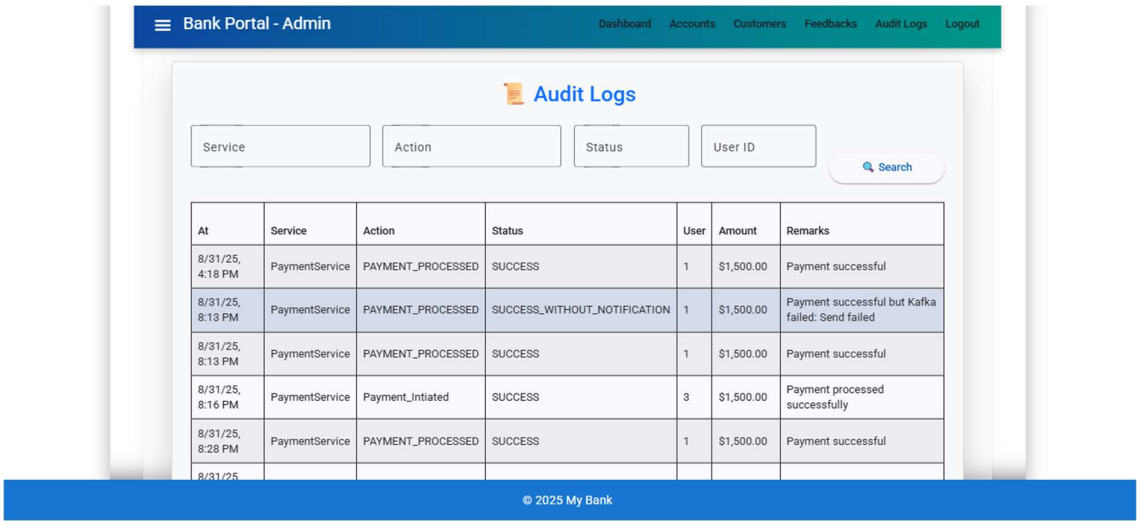
LOGIN PAGE:

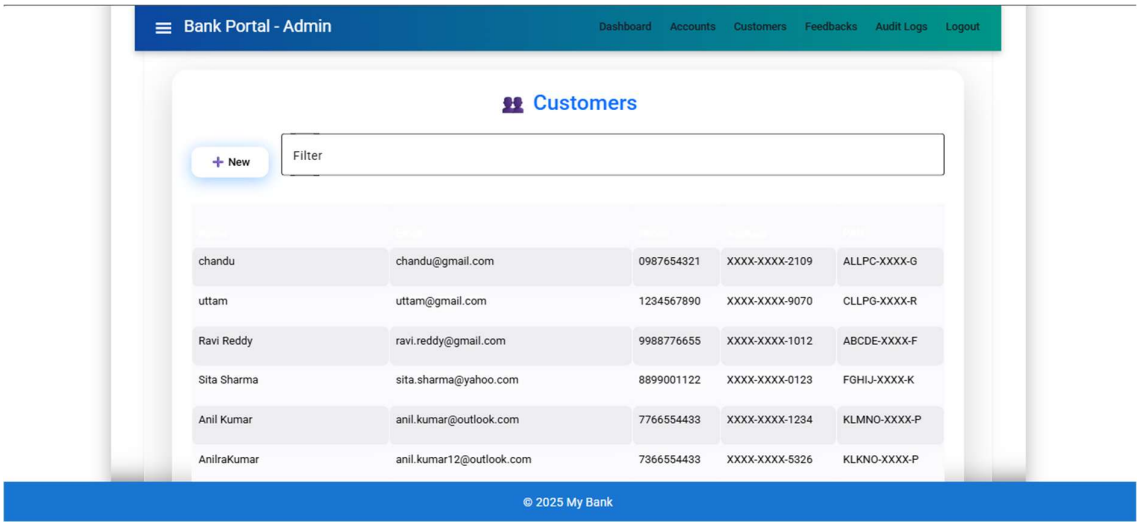


ADMIN COMPONENTS VIEWS:

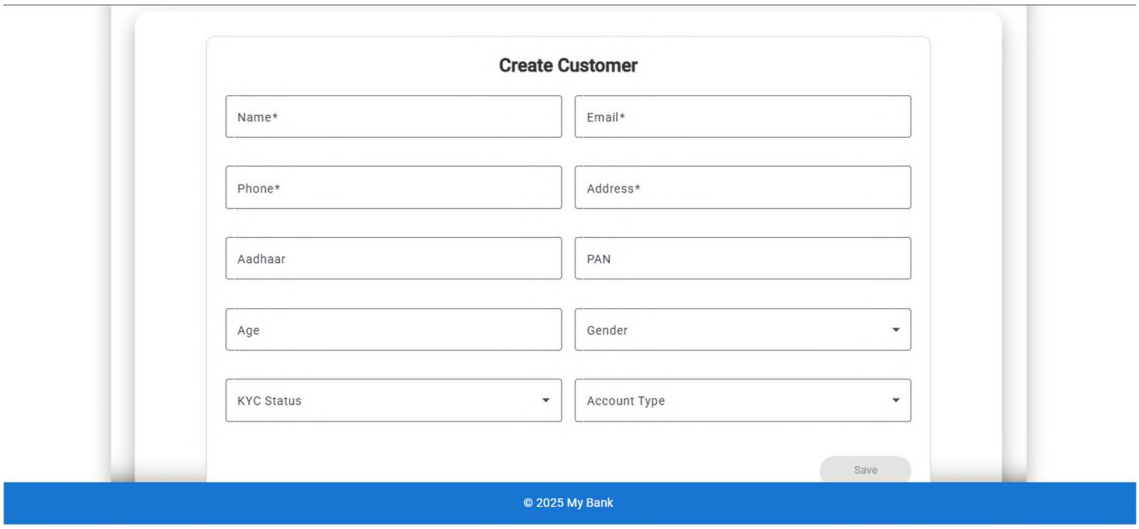


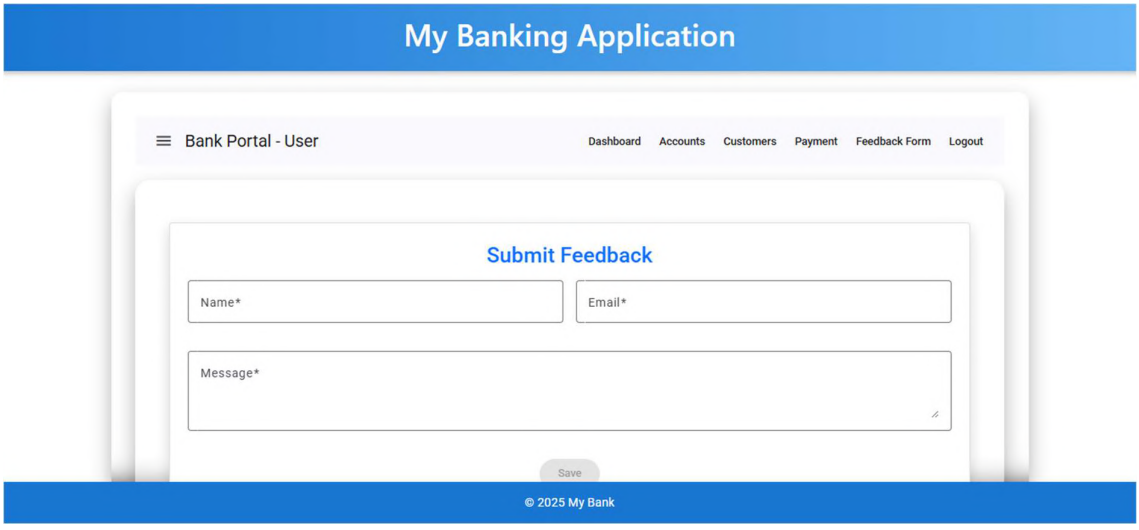


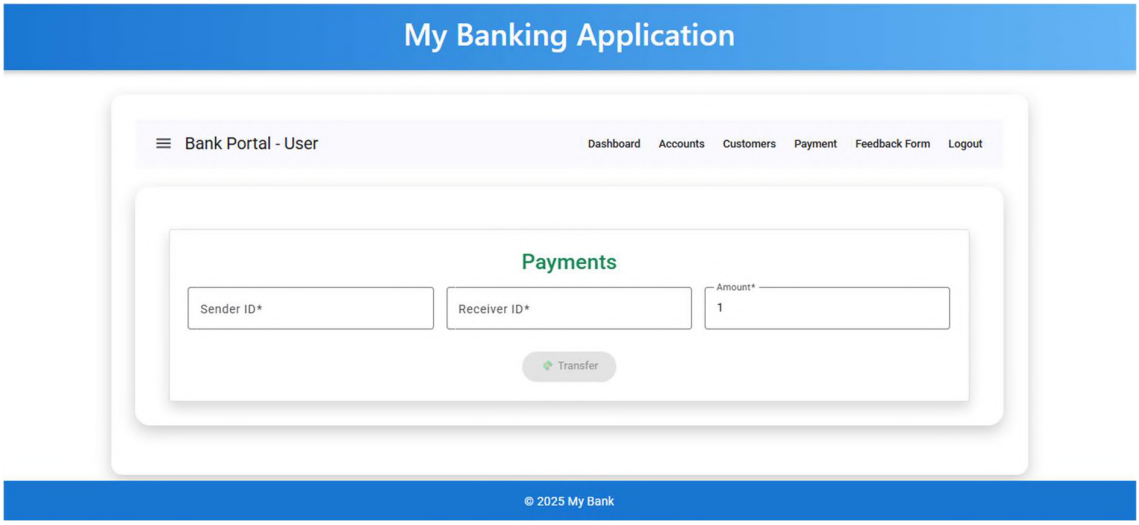




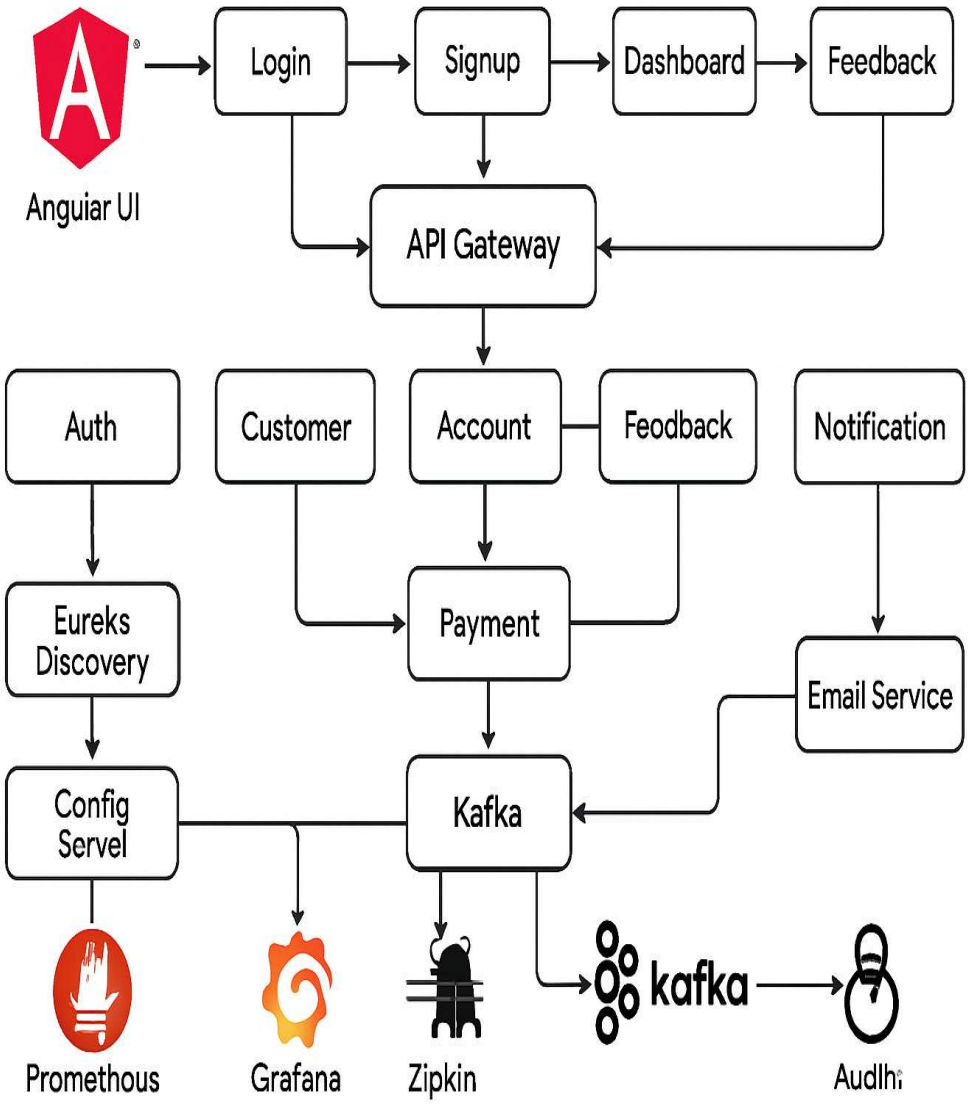
USER COMPONENTS VIEWS:





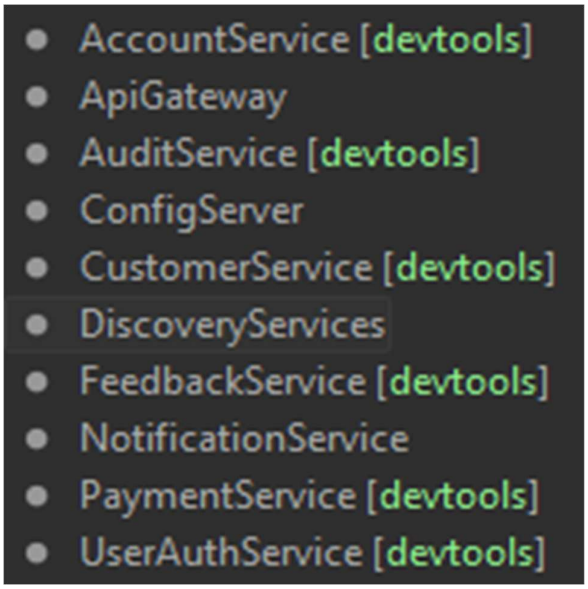


ARCHITECTURE:



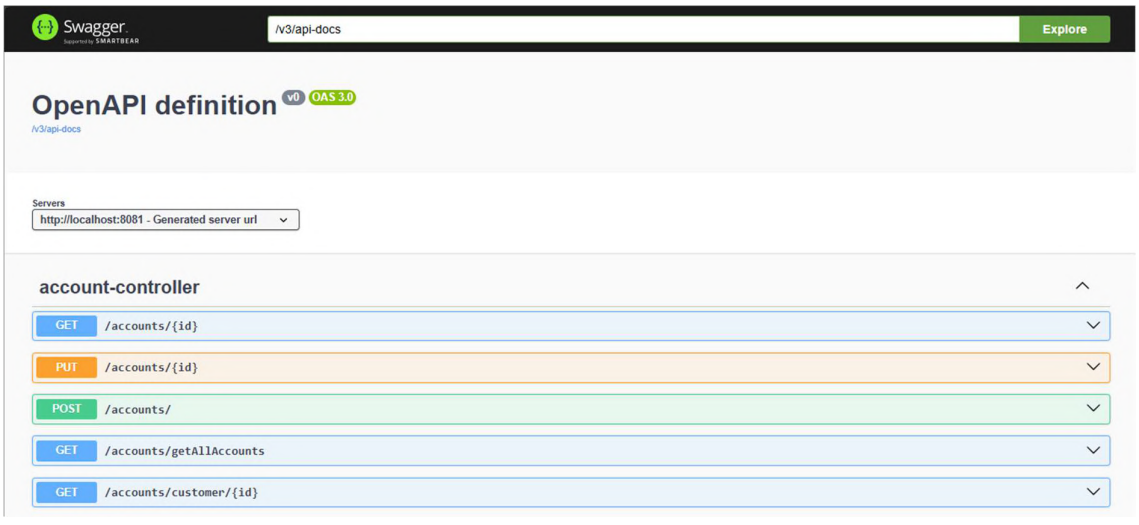
JAVA-SPRING_BOOT_MICROSERVICES:

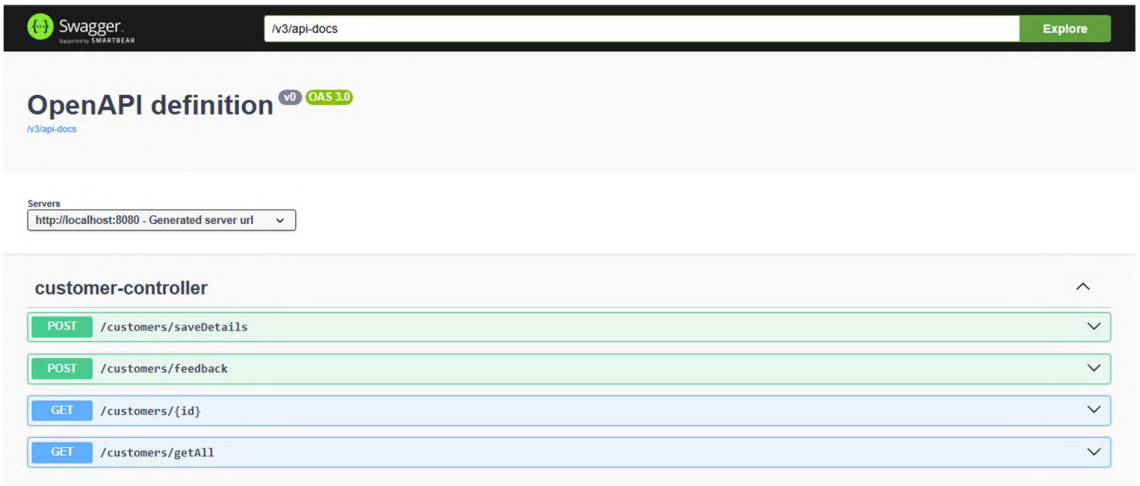
SERVICES:

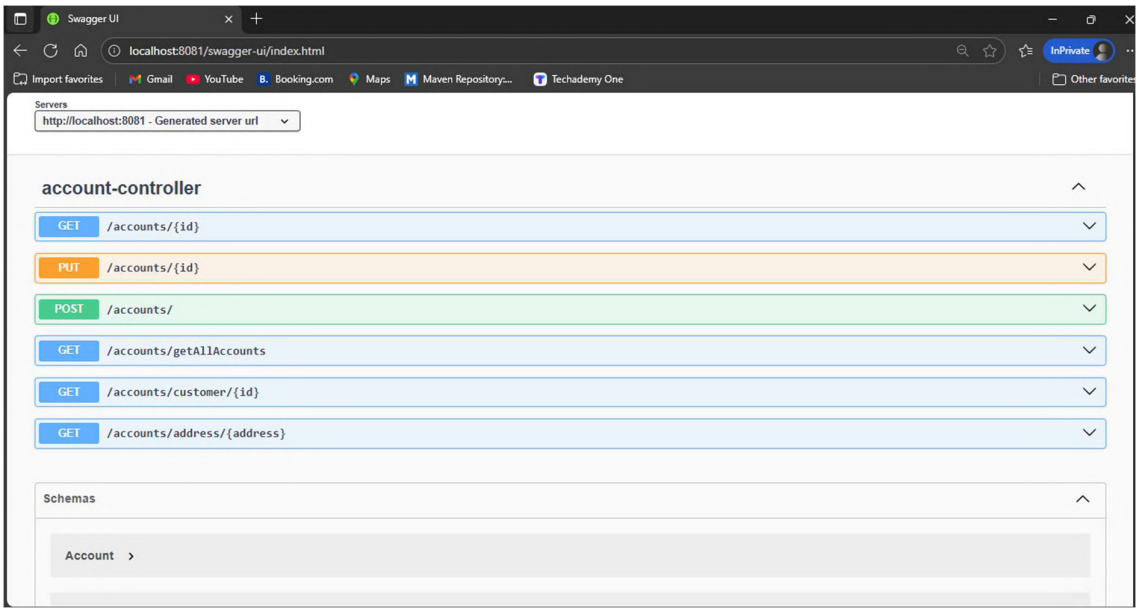


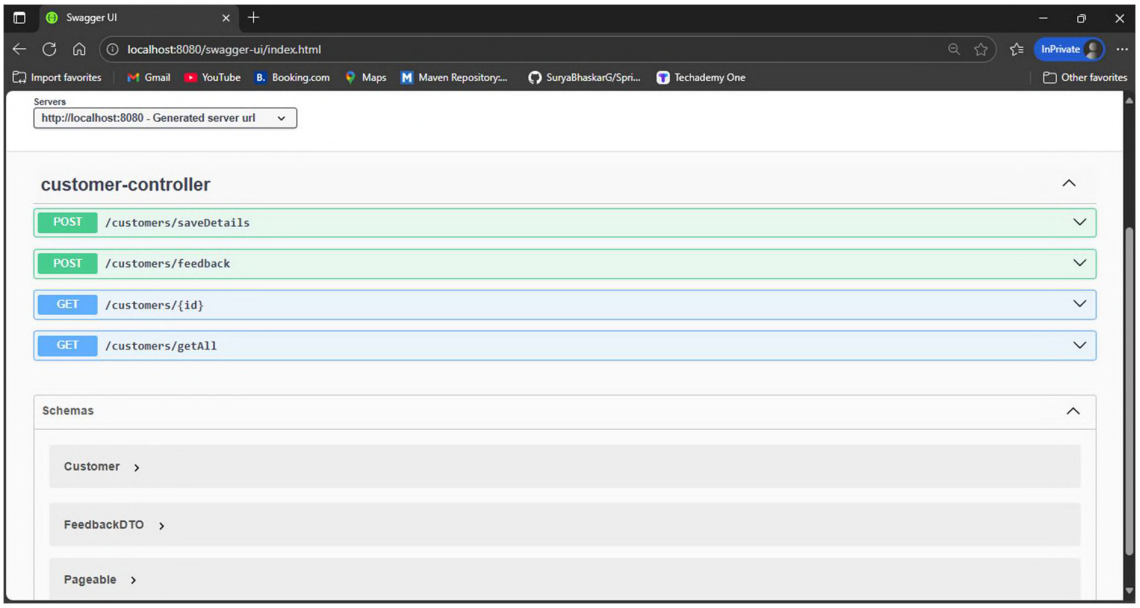
Service	Responsibility	Database
API Gateway	Routes client requests, handles load balancing	N/A
AuthService	User authentication, JWT issuance	AuthDB
AccountService	Account data and balance management	AccountDB
CustomerService	Customer profile and KYC data	CustomerDB
PaymentService	Handles payments, transaction workflow	PaymentDB
FeedbackService	User feedback storage and reporting	FeedbackDB
NotificationService	Async notifications via Kafka	NotificationDB
AuditService	Transaction logs, audit trails	AuditDB

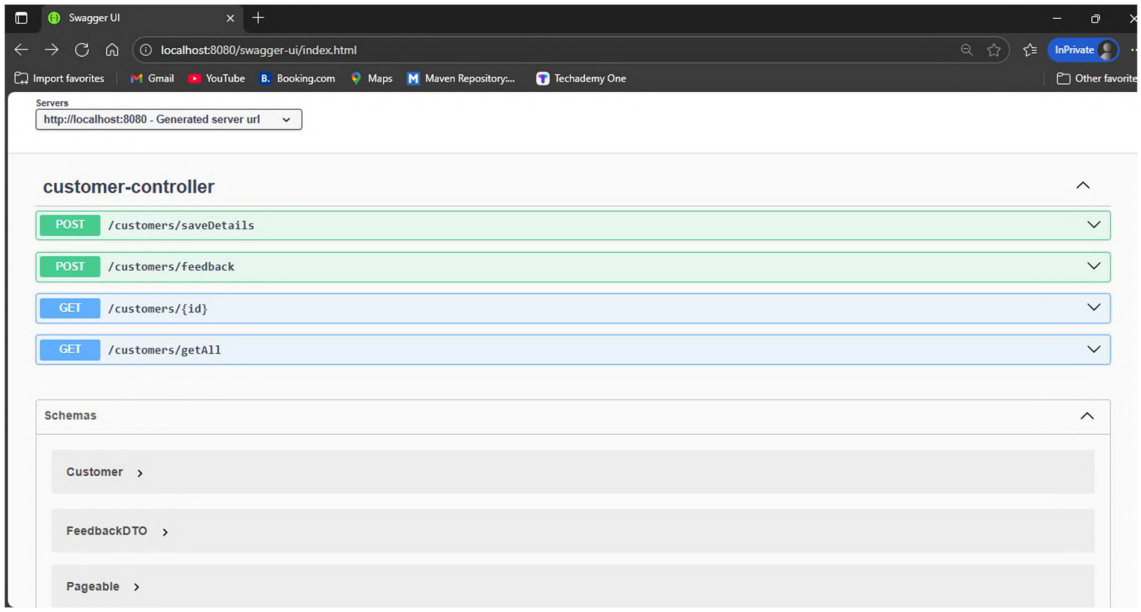
SWAGGER and ZIPKIN and PROMETHEUS and GRAFANA :

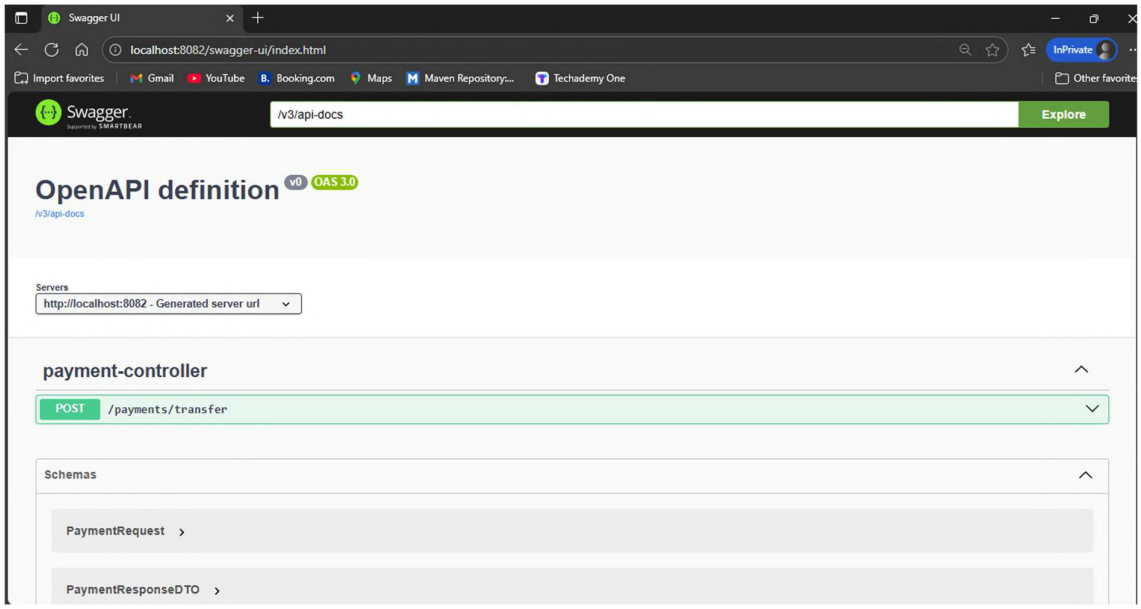


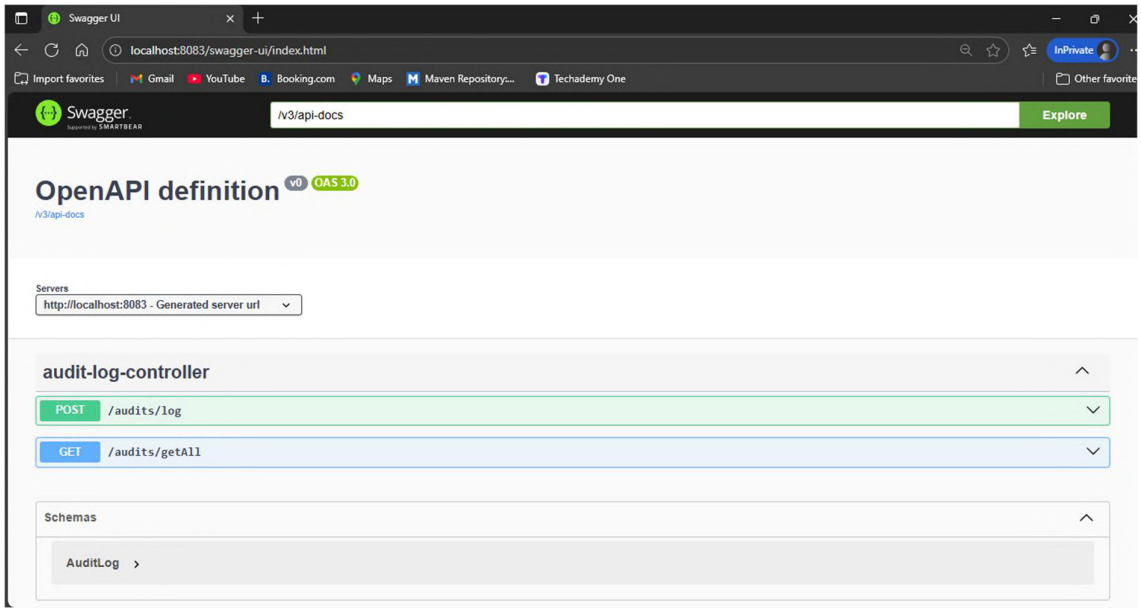

Swagger
/v3/api-docs
Explore
OpenAPI definition
v0
OAS 3.0
/v3/api-docs
Servers
http://localhost:8080 - Generated server url
customer-controller
POST /customers/saveDetails
POST /customers/feedback
GET /customers/{id}
GET /customers/getAll

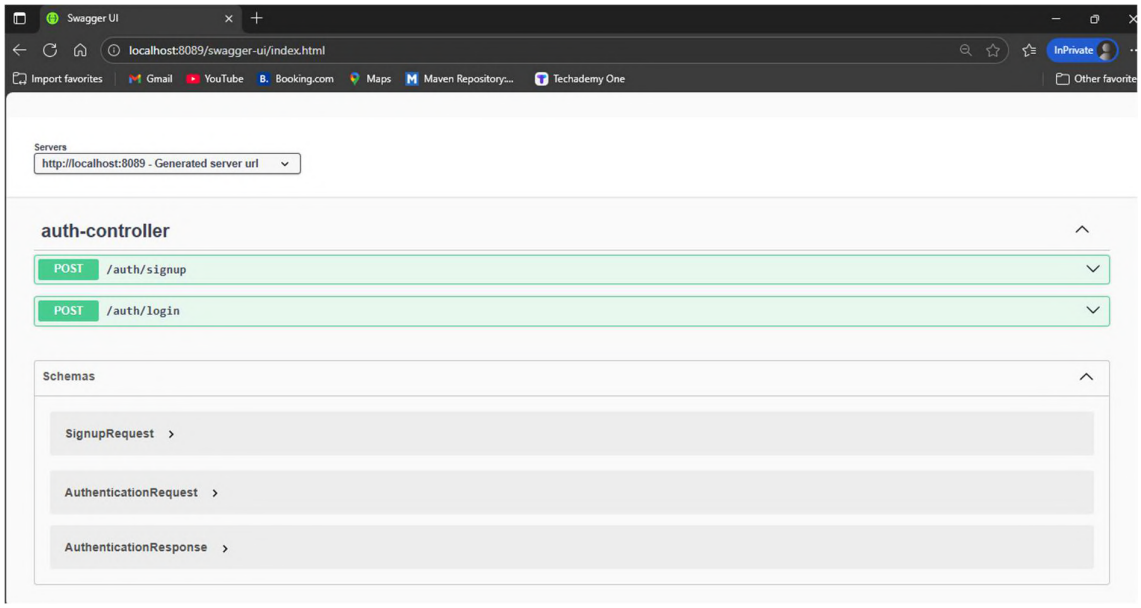

Swagger UI
localhost:8081/swagger-ui/index.html
InPrivate
Import favorites
Gmail
YouTube
Booking.com
Maps
Maven Repository:...
Techademy One
Other favorites
Servers
http://localhost:8081 - Generated server url
account-controller
GET /accounts/{id}
PUT /accounts/{id}
POST /accounts/
GET /accounts/getAllAccounts
GET /accounts/customer/{id}
GET /accounts/address/{address}
Schemas
Account

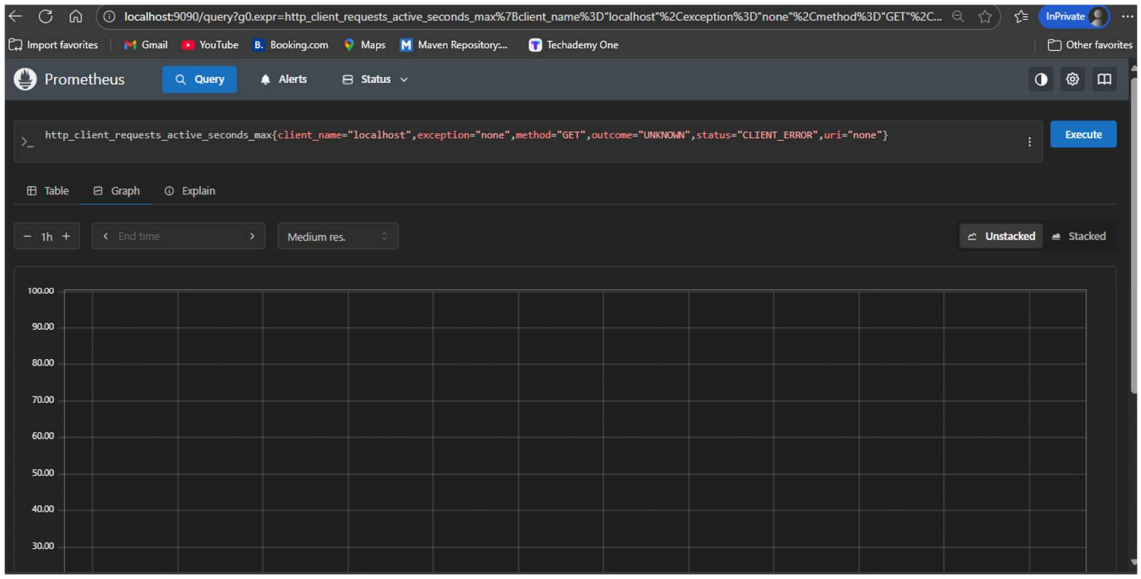

Swagger UI
localhost:8080/swagger-ui/index.html
InPrivate
Import favorites
Gmail
YouTube
Booking.com
Maps
Maven Repository:...
SuryaBhaskarG/Spri...
Techademy One
Other favorites
Servers
http://localhost:8080 - Generated server url
customer-controller
POST /customers/saveDetails
POST /customers/feedback
GET /customers/{id}
GET /customers/getAll
Schemas
Customer
FeedbackDTO
Pageable

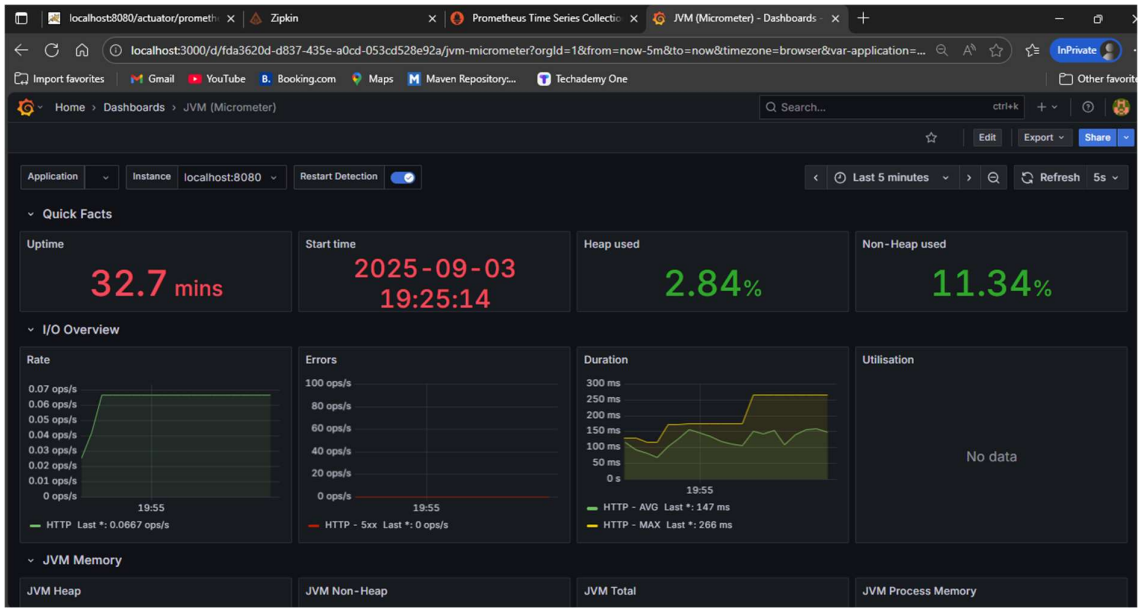

Swagger UI
localhost:8080/swagger-ui/index.html
Servers
http://localhost:8080 - Generated server url
customer-controller
POST /customers/saveDetails
POST /customers/feedback
GET /customers/{id}
GET /customers/getAll
Schemas
Customer
FeedbackDTO
Pageable

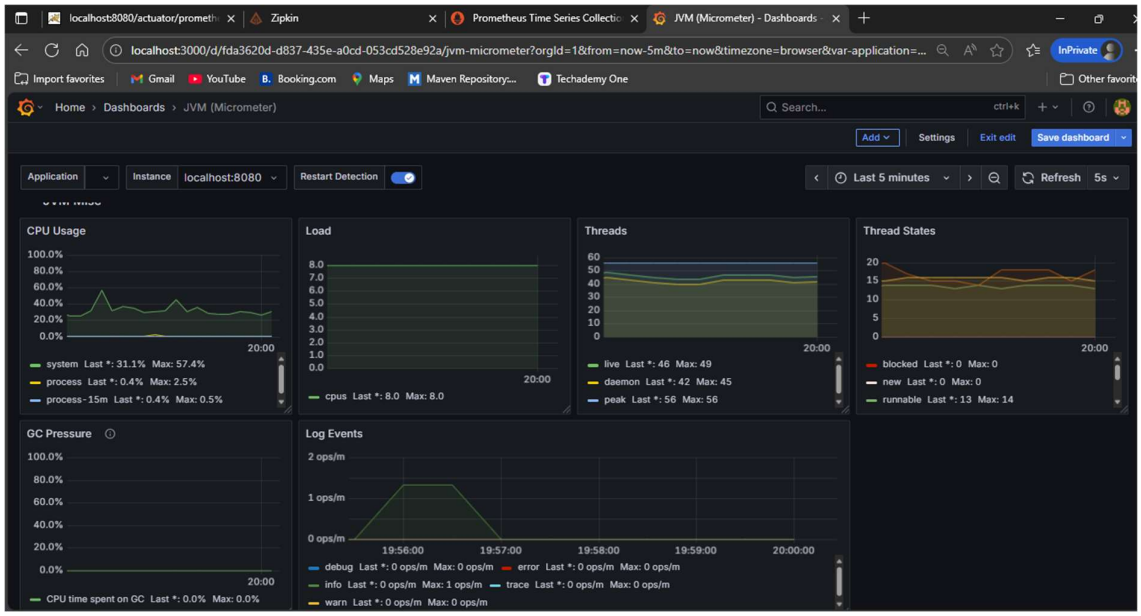

Swagger UI
localhost:8082/swagger-ui/index.html
Swagger
/v3/api-docs
Explore
OpenAPI definition
v0
OAS 3.0
/v3/api-docs
Servers
http://localhost:8082 - Generated server url
payment-controller
POST /payments/transfer
Schemas
PaymentRequest
PaymentResponseDTO

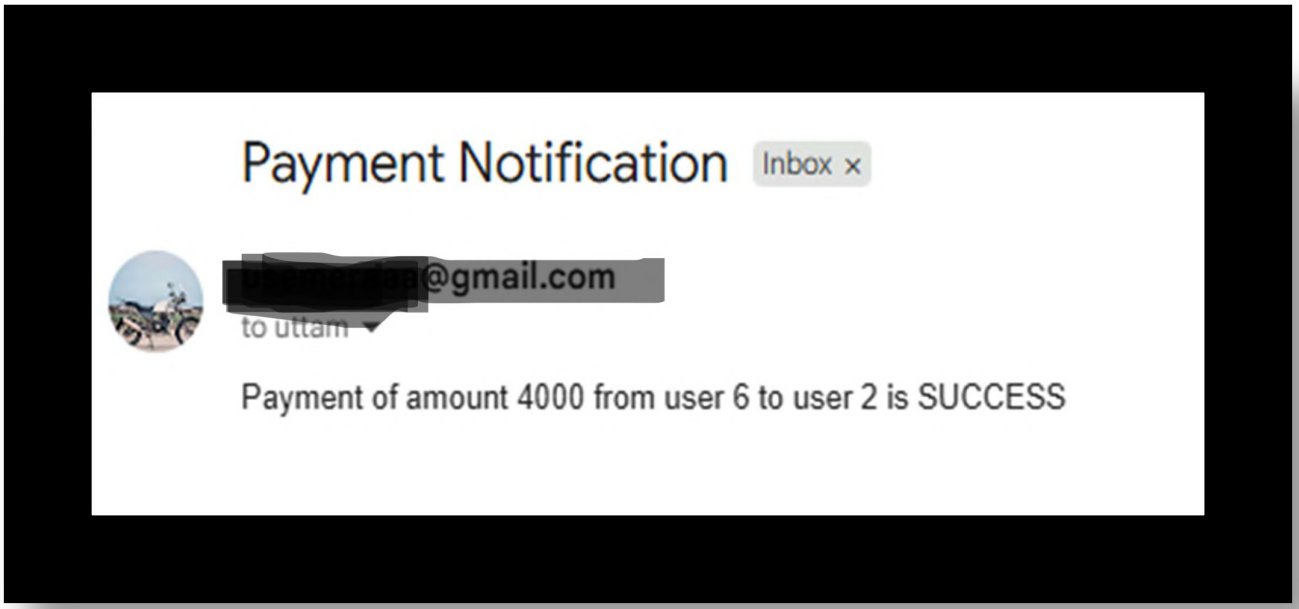

Swagger UI
localhost:8083/swagger-ui/index.html
Swagger
/v3/api-docs
Explore
OpenAPI definition
v0
OAS 3.0
/v3/api-docs
Servers
http://localhost:8083 - Generated server url
audit-log-controller
POST /audits/log
GET /audits/getAll
Schemas
AuditLog


Swagger UI
localhost:8089/swagger-ui/index.html
Servers
http://localhost:8089 - Generated server url
auth-controller
POST /auth/signup
POST /auth/login
Schemas
SignupRequest
AuthenticationRequest
AuthenticationResponse


Prometheus
Query
Alerts
Status
http_client_requests_active_seconds_max{client_name="localhost",exception="none",method="GET",outcome="UNKNOWN",status="CLIENT_ERROR",uri="none"}
Execute
Table
Graph
Explain
Unstacked
Stacked
100.00
90.00
80.00
70.00
60.00
50.00
40.00
30.00


JVM (Micrometer) - Dashboards
Home › Dashboards › JVM (Micrometer)
Application
Instance localhost:8080
Restart Detection
Last 5 minutes
Refresh 5s
Quick Facts
Uptime
32.7 mins
Start time
2025-09-03 19:25:14
Heap used
2.84%
Non-Heap used
11.34%
I/O Overview
Rate
Errors
Duration
Utilisation
No data
JVM Memory
JVM Heap
JVM Non-Heap
JVM Total
JVM Process Memory


localhost8080/actuator/prometh
Zipkin
Prometheus Time Series Collectio
JVM (Micrometer) - Dashboards
localhost:3000/d/fda3620d-d837-435e-a0cd-053cd528e92a/jvm-micrometer?orgId=1&from=now-5m&to=now&timezone=browser&var-application=...
InPrivate
Import favorites
Gmail
YouTube
Booking.com
Maps
Maven Repository:...
Techademy One
Other favorite
Home › Dashboards › JVM (Micrometer)
Search...
ctrl+k
Add
Settings
Exit edit
Save dashboard
Application
Instance
localhost:8080
Restart Detection
Last 5 minutes
Refresh
5s
CPU Usage
100.0%
80.0%
60.0%
40.0%
20.0%
0.0%
20:00
system Last *: 31.1% Max: 57.4%
process Last *: 0.4% Max: 2.5%
process-15m Last *: 0.4% Max: 0.5%
Load
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
20:00
cpus Last *: 8.0 Max: 8.0
Threads
60
50
40
30
20
10
0
20:00
live Last *: 46 Max: 49
daemon Last *: 42 Max: 45
peak Last *: 56 Max: 56
Thread States
20
15
10
5
0
20:00
blocked Last *: 0 Max: 0
new Last *: 0 Max: 0
runnable Last *: 13 Max: 14
GC Pressure
100.0%
80.0%
60.0%
40.0%
20.0%
0.0%
20:00
CPU time spent on GC Last *: 0.0% Max: 0.0%
Log Events
2 ops/m
1 ops/m
0 ops/m
19:56:00
19:57:00
19:58:00
19:59:00
20:00:00
debug Last *: 0 ops/m Max: 0 ops/m
error Last *: 0 ops/m Max: 0 ops/m
info Last *: 0 ops/m Max: 1 ops/m
trace Last *: 0 ops/m Max: 0 ops/m
warn Last *: 0 ops/m Max: 0 ops/m


Payment Notification
Inbox ×
@gmail.com
to uttam
Payment of amount 4000 from user 6 to user 2 is SUCCESS

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 **ANGULAR**