Banking Application

Backend Using microservices

Introduction :

In todays financial sector. Customers expect secure, reliable and real time banking services across multiple channels. Traditional monolithic systems often struggle to keep up with these expectations due to scalability limitations, tight coupling and maintenance challenges.

To overcome these issues, we have developed a banking application using microservices for backend. The banking application is designed as a secure, scalable and modular backend system for managing essential banking operations. This project adopts a microservices architecture to ensure flexibility, reliability and easy to maintain. By dividing the application into different services, each responsible for a specific banking domain, the system achieves loose coupling, high scalability and fault isolation.

The platform supports essential banking features such as:

- Customer onboarding and authentication
- Account creation and management (savings/current)
- Internal and external money transfers
- Transaction history and digital statements
- Real-time notifications (SMS/Email)
- Audit logging, compliance, and reporting

Technologies used :

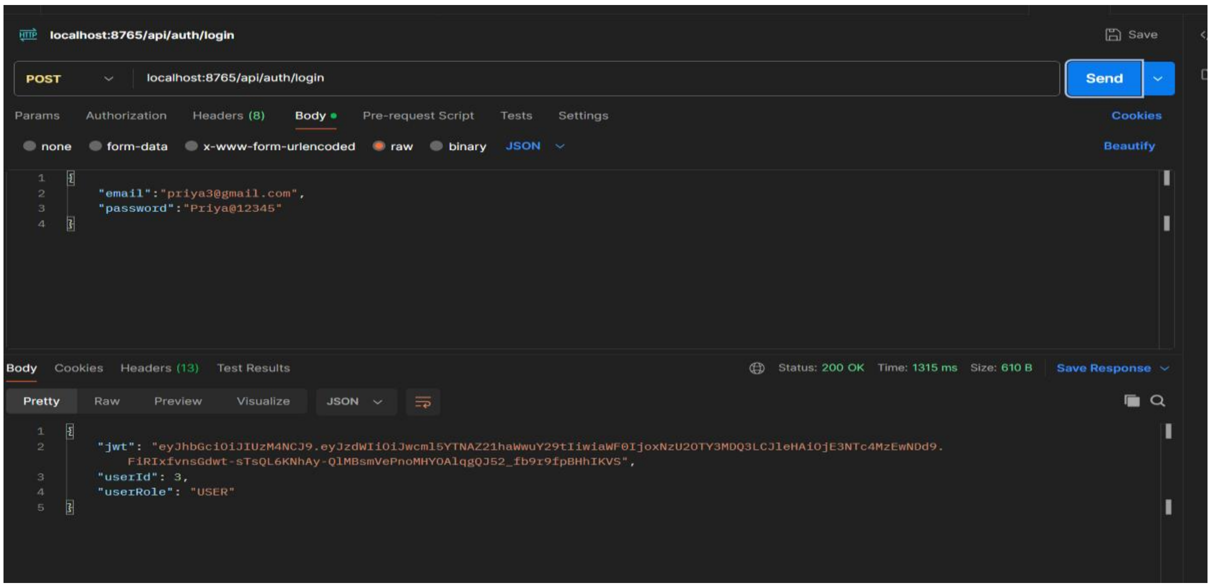
- ✓ 1.Spring boot, Spring Cloud for service development.
- ✓ 2.Eureka Server/Api-Gateway for service discovery and routing.
- ✓ 3.MySQl for data Storage.
- ✓ 4.Swagger UI for Api documentation.
- ✓ 5.Lombok, Security, JWT for authentication and faster development.

Authentication & Security

It is the backbone of any banking application. To ensure only authorized users can access the system, this project implements authentication and authorization mechanisms using JWT.

JWT(Json Web Token) : It is used for secure, stateless user sessions. Role based access control ensures different permissions for users and admins.

User-Authentication service:

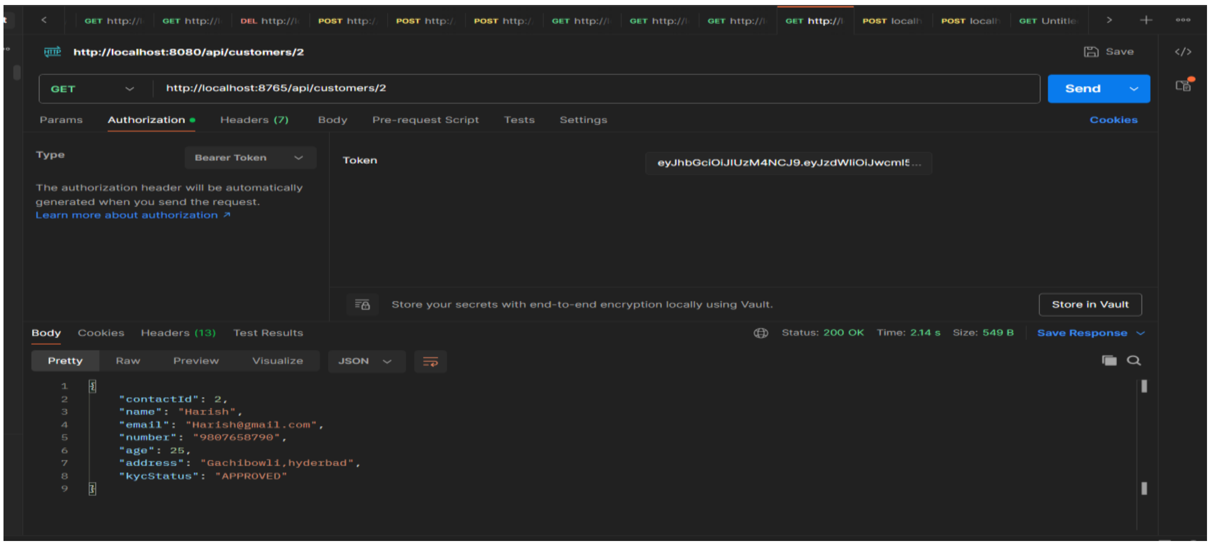

POST localhost:8765/api/auth/login

```
{
    "email":"priya3@gmail.com",
    "password":"Priya@12345"
}
```

Status: 200 OK Time: 1315 ms Size: 610 B

```
{
    "jwt": "eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJwcml5YTNAZ21haWwuY29tIiwiaWF0IjoxNzU2OTY3MDQ3LCJleHAiOjE3NTc4MzEwNDd9.
        FiRIxfvnsGdwt-sTsQL6KNhAy-QlMBsmVePnoMHYOAlqgQJ52_fb9r9fpBHhIKVS",
    "userId": 3,
    "userRole": "USER"
}
```

Jwt token is created successfully for login authentication.


GET http://localhost:8765/api/customers/2

Authorization: Type Bearer Token

Token eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJwcml5…

The authorization header will be automatically generated when you send the request.

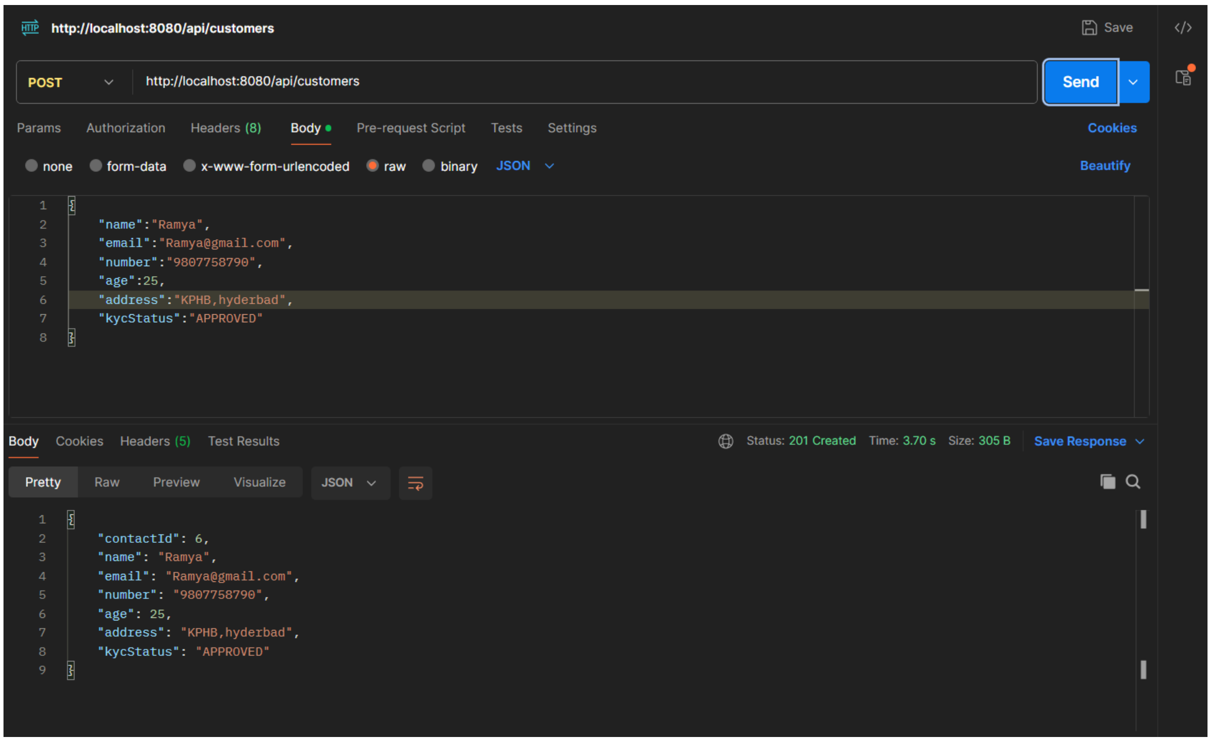
Status: 200 OK Time: 214 s Size: 549 B

```
{
    "contactId": 2,
    "name": "Harish",
    "email": "Harish@gmail.com",
    "number": "9807658790",
    "age": 25,
    "address": "Gachibowli,hyderbad",
    "kycStatus": "APPROVED"
}
```

After passing jwt token, we can access other services.

Customer Service :

Adding customers/users.


http://localhost:8080/api/customers

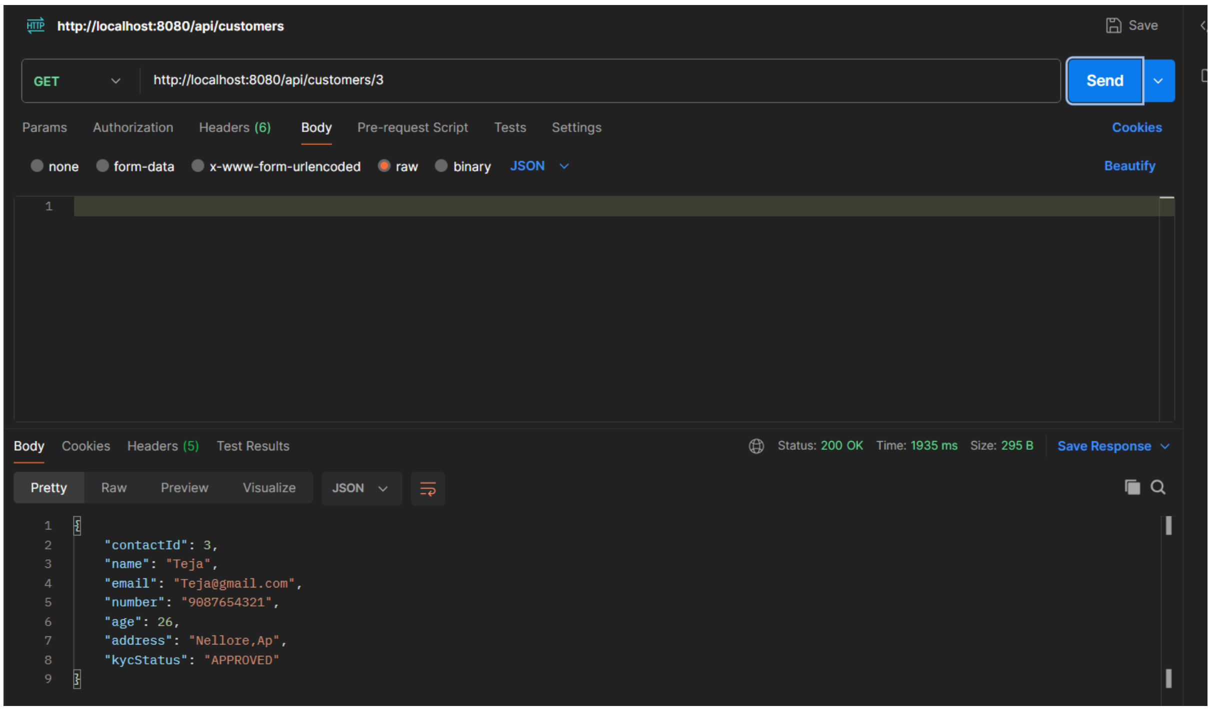
POST http://localhost:8080/api/customers

```
{
    "name":"Ramya",
    "email":"Ramya@gmail.com",
    "number":"9807758790",
    "age":25,
    "address":"KPHB,hyderbad",
    "kycStatus":"APPROVED"
}
```

Status: 201 Created Time: 3.70 s Size: 305 B

```
{
    "contactId": 6,
    "name": "Ramya",
    "email": "Ramya@gmail.com",
    "number": "9807758790",
    "age": 25,
    "address": "KPHB,hyderbad",
    "kycStatus": "APPROVED"
}
```

Getting customers by Id :


http://localhost:8080/api/customers

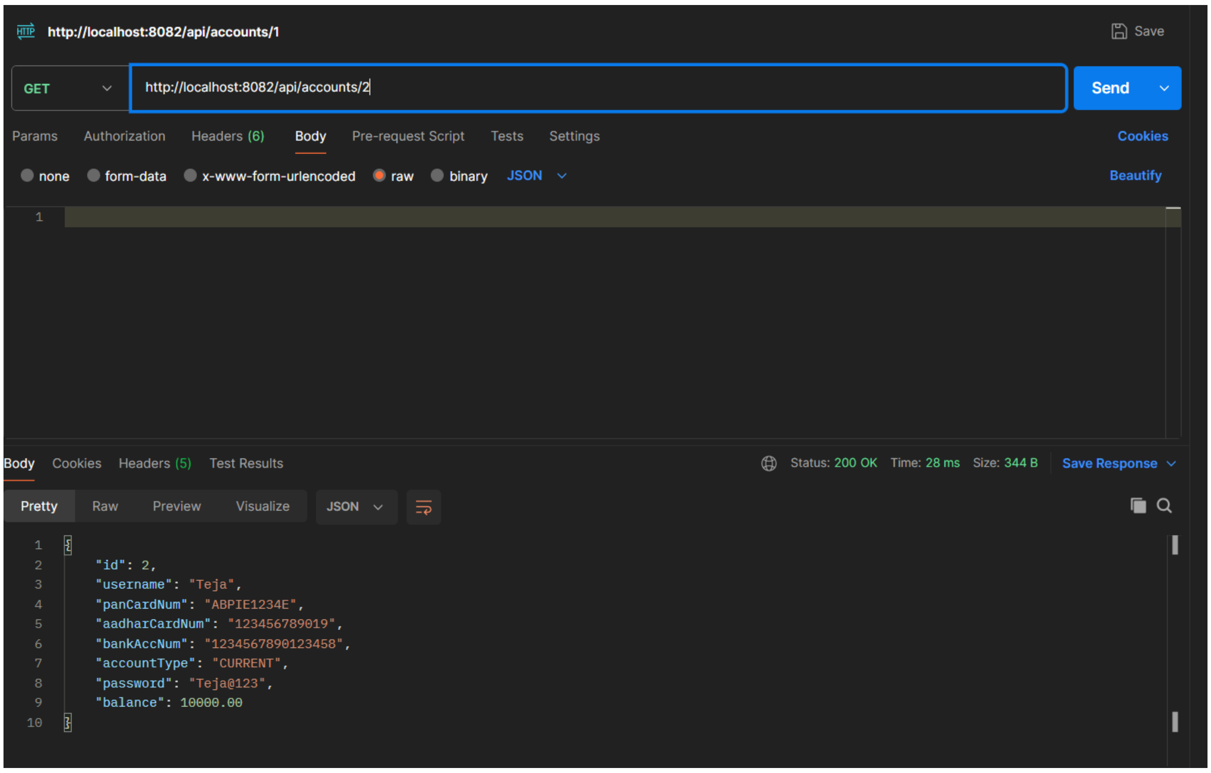
GET http://localhost:8080/api/customers/3

Status: 200 OK Time: 1935 ms Size: 295 B

```
{
    "contactId": 3,
    "name": "Teja",
    "email": "Teja@gmail.com",
    "number": "9087654321",
    "age": 26,
    "address": "Nellore,Ap",
    "kycStatus": "APPROVED"
}
```

Account Service :

Getting Account by Id

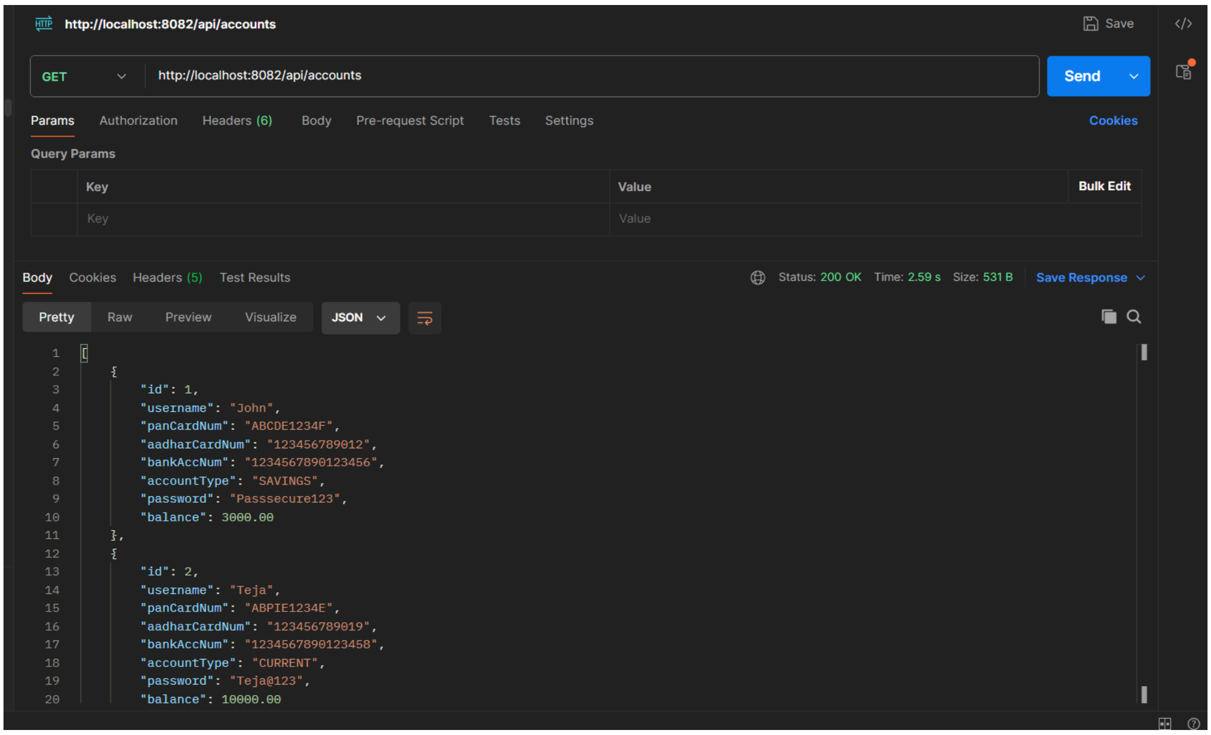


GET http://localhost:8082/api/accounts/2

Status: 200 OK Time: 28 ms Size: 344 B

```
{
    "id": 2,
    "username": "Teja",
    "panCardNum": "ABPIE1234E",
    "aadharCardNum": "123456789019",
    "bankAccNum": "1234567890123458",
    "accountType": "CURRENT",
    "password": "Teja@123",
    "balance": 10000.00
}
```

Getting All Accounts

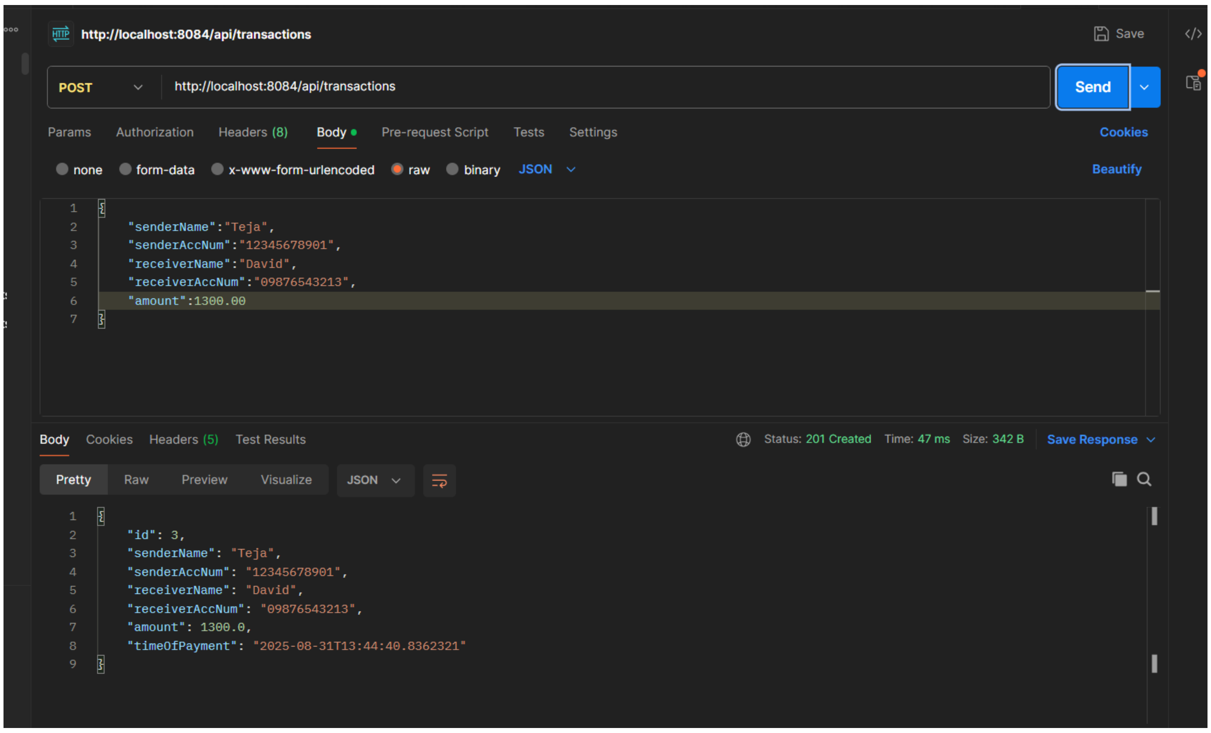


GET http://localhost:8082/api/accounts

Status: 200 OK Time: 2.59 s Size: 531 B

```
[
    {
        "id": 1,
        "username": "John",
        "panCardNum": "ABCDE1234F",
        "aadharCardNum": "123456789012",
        "bankAccNum": "1234567890123456",
        "accountType": "SAVINGS",
        "password": "Passsecure123",
        "balance": 3000.00
    },
    {
        "id": 2,
        "username": "Teja",
        "panCardNum": "ABPIE1234E",
        "aadharCardNum": "123456789019",
        "bankAccNum": "1234567890123458",
        "accountType": "CURRENT",
        "password": "Teja@123",
        "balance": 10000.00
```

Transaction Service :



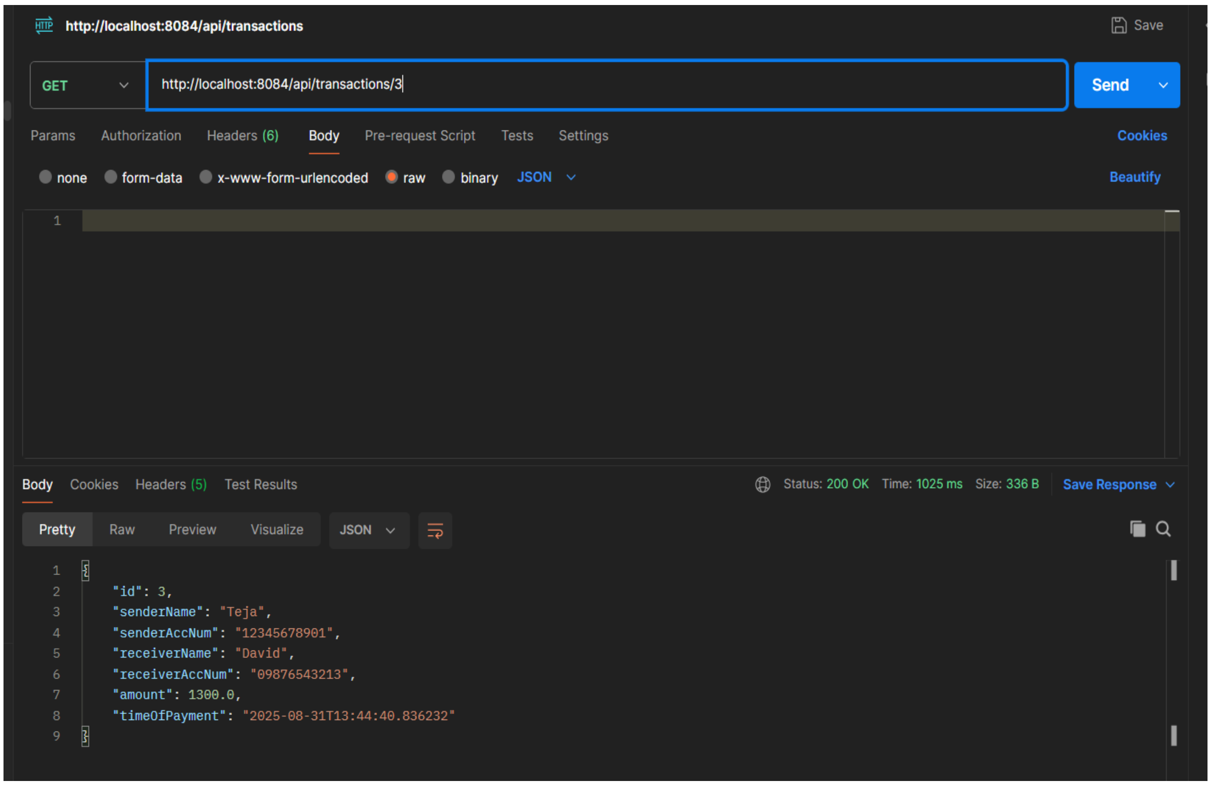
POST http://localhost:8084/api/transactions

```
{
    "senderName":"Teja",
    "senderAccNum":"12345678901",
    "receiverName":"David",
    "receiverAccNum":"09876543213",
    "amount":1300.00
}
```

Status: 201 Created Time: 47 ms Size: 342 B

```
{
    "id": 3,
    "senderName": "Teja",
    "senderAccNum": "12345678901",
    "receiverName": "David",
    "receiverAccNum": "09876543213",
    "amount": 1300.0,
    "timeOfPayment": "2025-08-31T13:44:40.8362321"
}
```

Getting transaction by Id

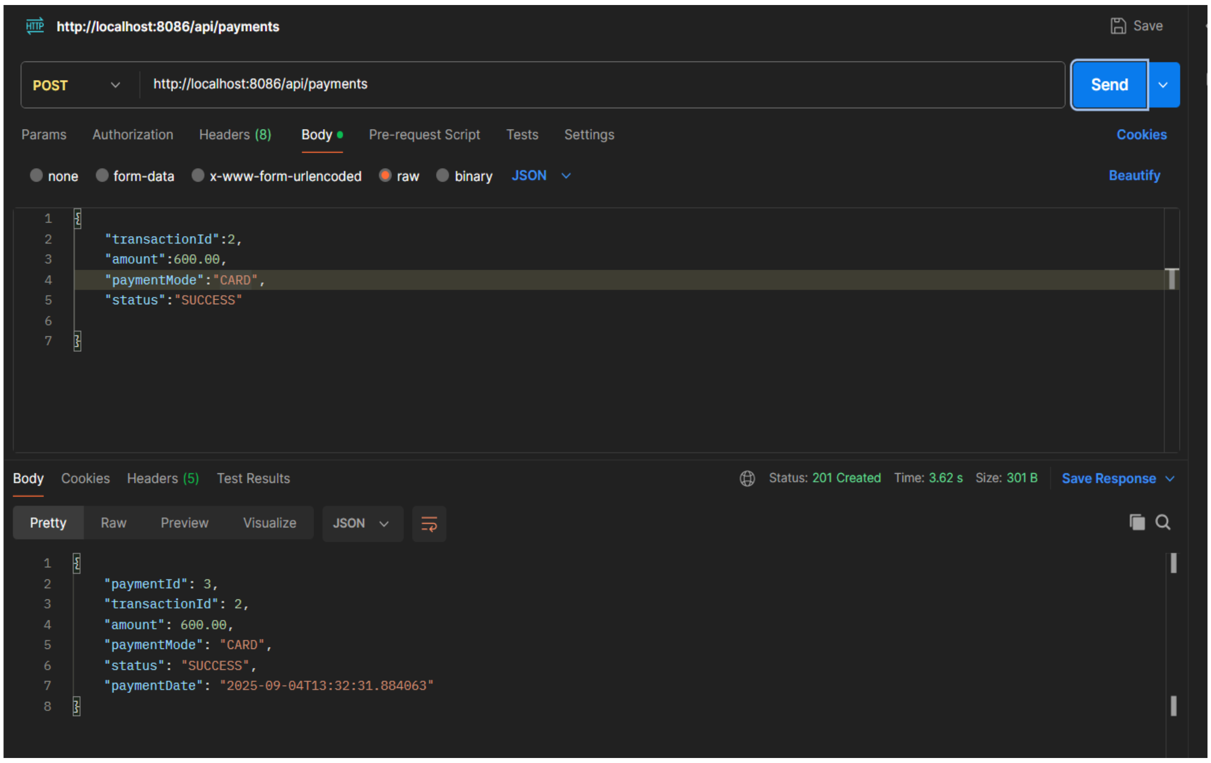


GET http://localhost:8084/api/transactions/3

Status: 200 OK Time: 1025 ms Size: 336 B

```
{
    "id": 3,
    "senderName": "Teja",
    "senderAccNum": "12345678901",
    "receiverName": "David",
    "receiverAccNum": "09876543213",
    "amount": 1300.0,
    "timeOfPayment": "2025-08-31T13:44:40.836232"
}
```

Payment Service

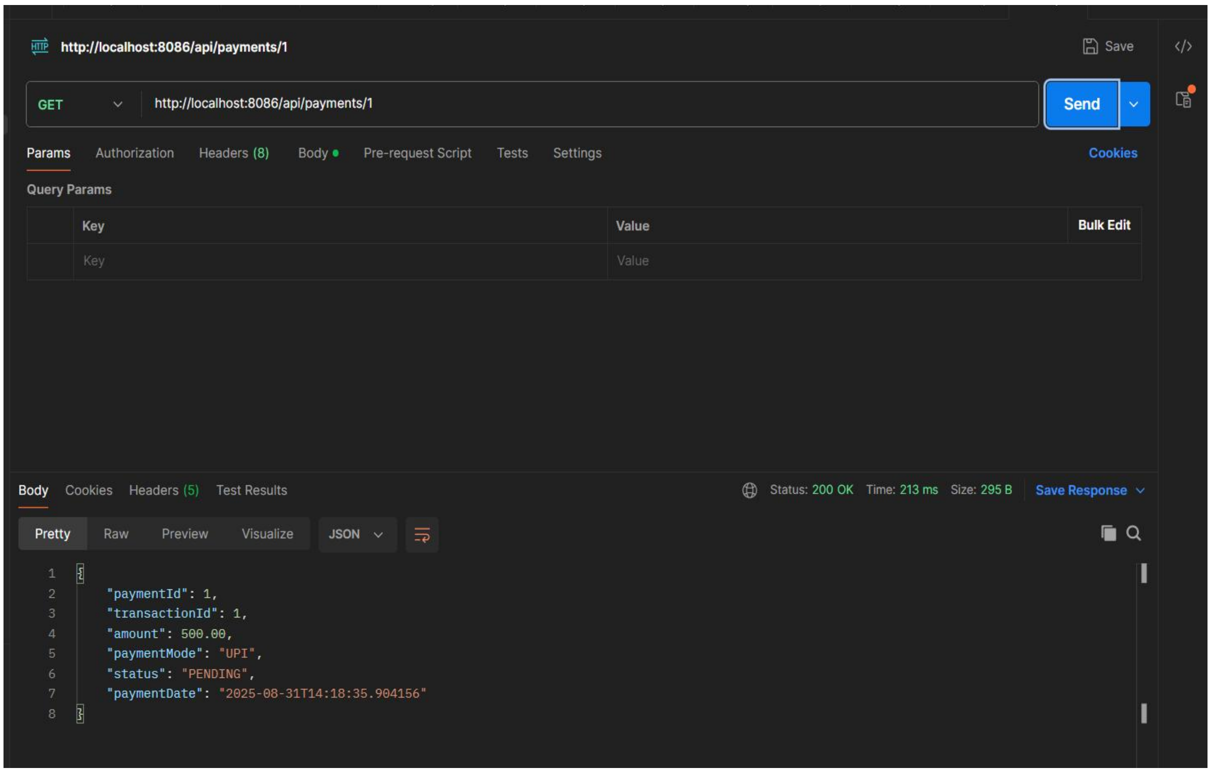


```
{
    "transactionId":2,
    "amount":600.00,
    "paymentMode":"CARD",
    "status":"SUCCESS"

}
```

```
{
    "paymentId": 3,
    "transactionId": 2,
    "amount": 600.00,
    "paymentMode": "CARD",
    "status": "SUCCESS",
    "paymentDate": "2025-09-04T13:32:31.884063"
}
```

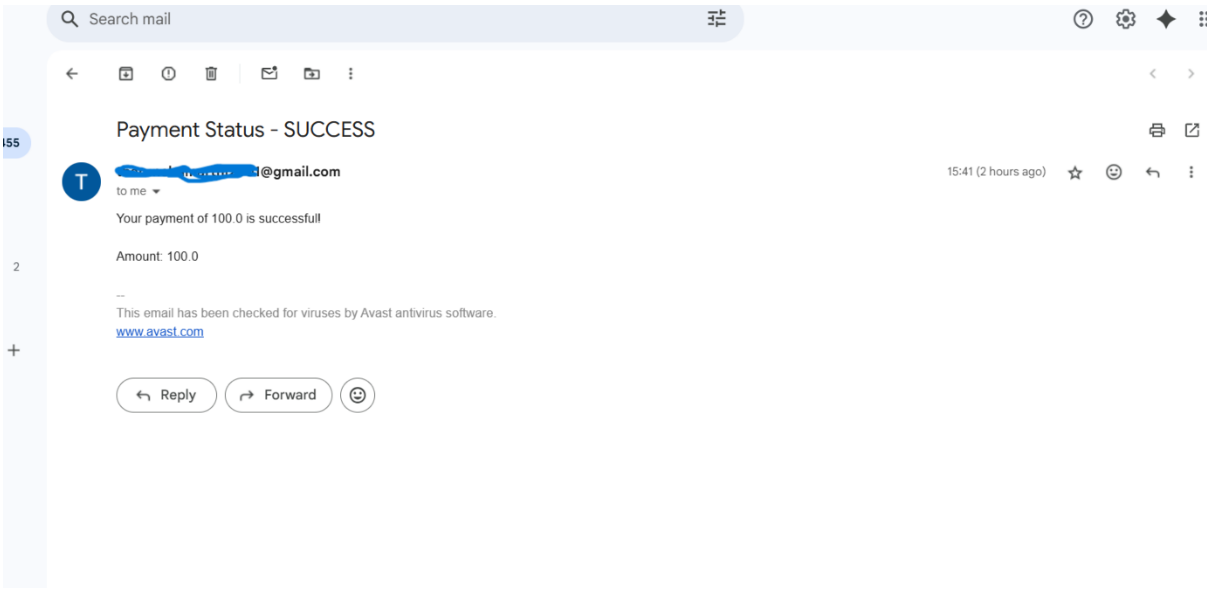
Getting payments by Id



```
{
    "paymentId": 1,
    "transactionId": 1,
    "amount": 500.00,
    "paymentMode": "UPI",
    "status": "PENDING",
    "paymentDate": "2025-08-31T14:18:35.904156"
}
```

Using asynchronous communication (Kafka) :

Kafka --- It is used instead of direct service-to-service calls. critical events (e.g., money transfer).It is also called as Event-Driven communication. It uses scalability, reliability and performance.



Payment Status - SUCCESS

[redacted]d@gmail.com
to me

15:41 (2 hours ago)

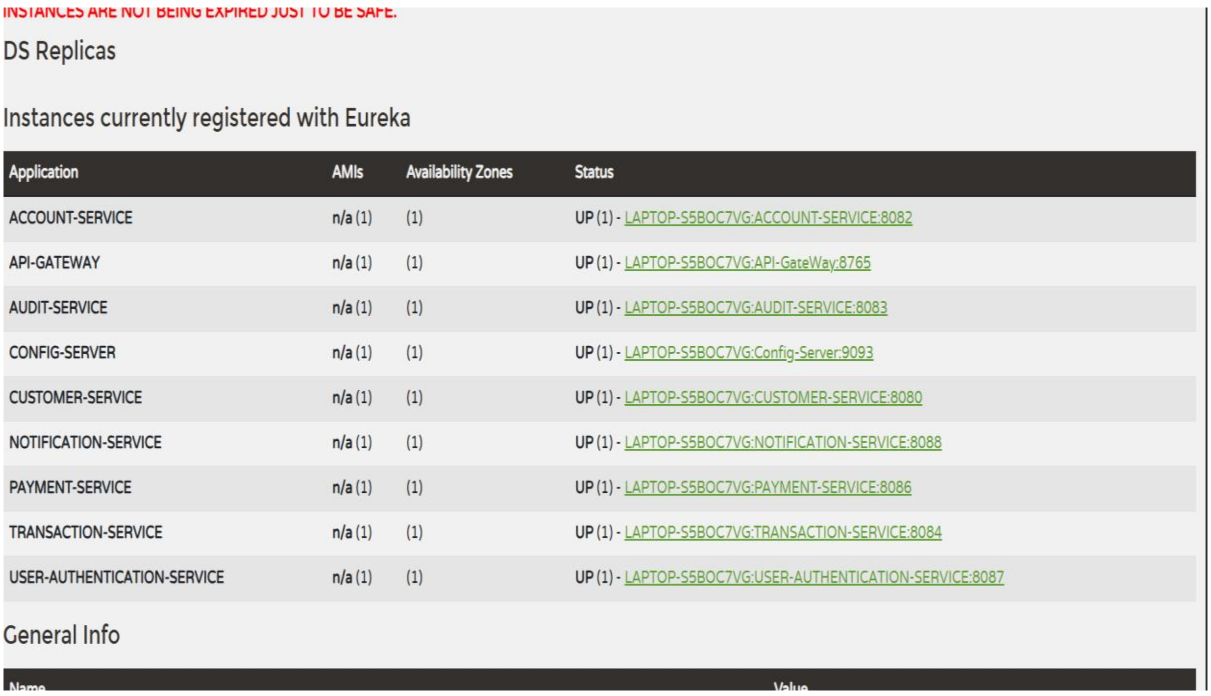
Your payment of 100.0 is successfull

Amount: 100.0

--
This email has been checked for viruses by Avast antivirus software.
www.avast.com

Eureka Server

All my services are registered with eureka server and passing through Api-gateway.



INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

Instances currently registered with Eureka

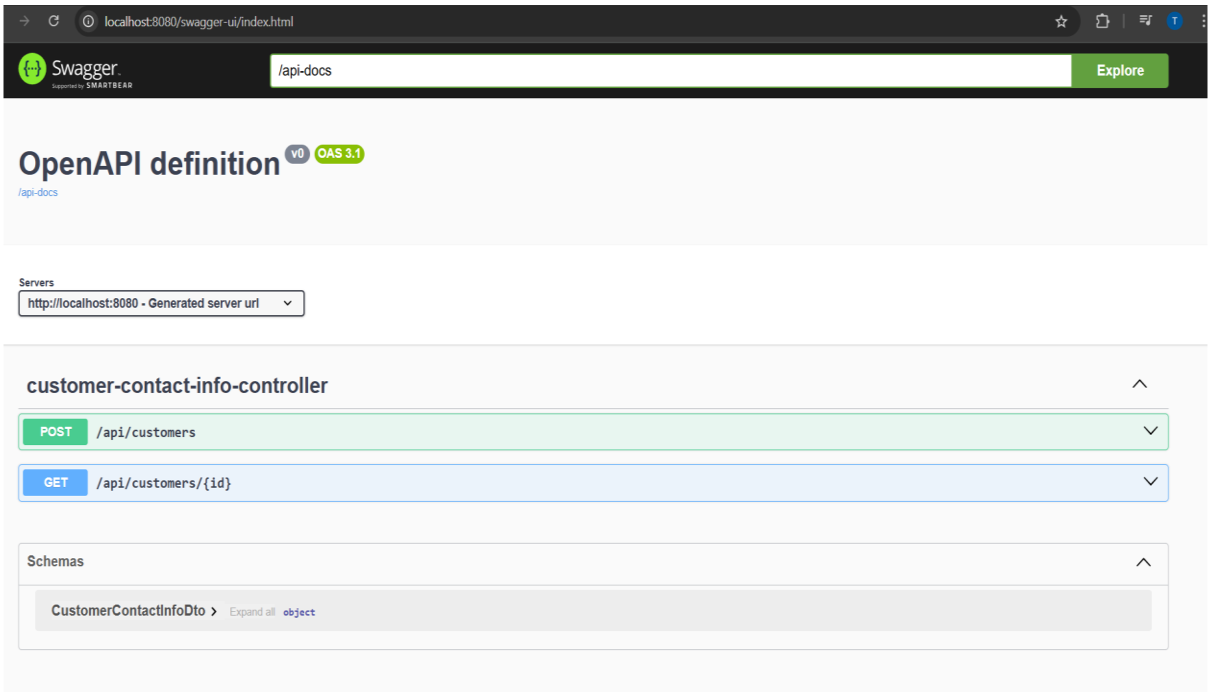
Application	AMIs	Availability Zones	Status
ACCOUNT-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-S5BOC7VG:ACCOUNT-SERVICE:8082
API-GATEWAY	n/a (1)	(1)	UP (1) - LAPTOP-S5BOC7VG:API-GateWay:8765
AUDIT-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-S5BOC7VG:AUDIT-SERVICE:8083
CONFIG-SERVER	n/a (1)	(1)	UP (1) - LAPTOP-S5BOC7VG:Config-Server:9093
CUSTOMER-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-S5BOC7VG:CUSTOMER-SERVICE:8080
NOTIFICATION-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-S5BOC7VG:NOTIFICATION-SERVICE:8088
PAYMENT-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-S5BOC7VG:PAYMENT-SERVICE:8086
TRANSACTION-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-S5BOC7VG:TRANSACTION-SERVICE:8084
USER-AUTHENTICATION-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-S5BOC7VG:USER-AUTHENTICATION-SERVICE:8087

General Info

Name	Value

Swagger-UI

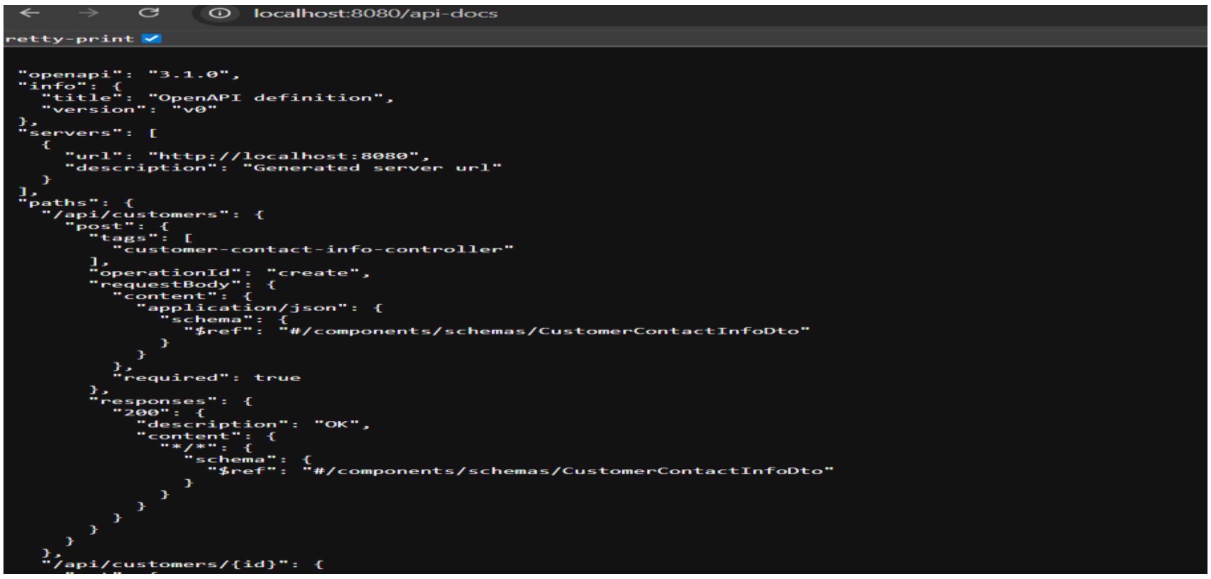
It is mainly used to make the system developer-friendly and transparent. It provides an easy-to-use interface where developers and testers can explore endpoints, view request/response models and execute API calls directly from the browser.



Api-Docs (Open-Api Specification)

By combining both Swagger-UI and Api-Docs, the system achieves

seamless frontend-backend integration .



```
"openapi": "3.1.0",
"info": {
  "title": "OpenAPI definition",
  "version": "v0"
},
"servers": [
  {
    "url": "http://localhost:8080",
    "description": "Generated server url"
  }
],
"paths": {
  "/api/customers": {
    "post": {
      "tags": [
        "customer-contact-info-controller"
      ],
      "operationId": "create",
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/CustomerContactInfoDto"
            }
          }
        },
        "required": true
      },
      "responses": {
        "200": {
          "description": "OK",
          "content": {
            "*/*": {
              "schema": {
                "$ref": "#/components/schemas/CustomerContactInfoDto"
              }
            }
          }
        }
      }
    }
  },
  "/api/customers/{id}": {
```