Project Documentation: Banking Application using Spring Boot Microservices, React and Cloud

Table of Contents:

1. Introduction:

This document provides a detailed overview of the Spring Boot Microservices and React-based Banking Application project. It describes the architecture, setup instructions, implementation details, and deployment process.

2. Project Overview:

The Banking Application aims to provide a secure and efficient banking system that allows users to perform various banking operations. The system consists of multiple microservices built using Spring Boot and a React-based frontend. The microservices communicate with each other via RESTful APIs and interact with databases for data storage and retrieval.

3. Technology Stack:

The technology stack used in this project includes:

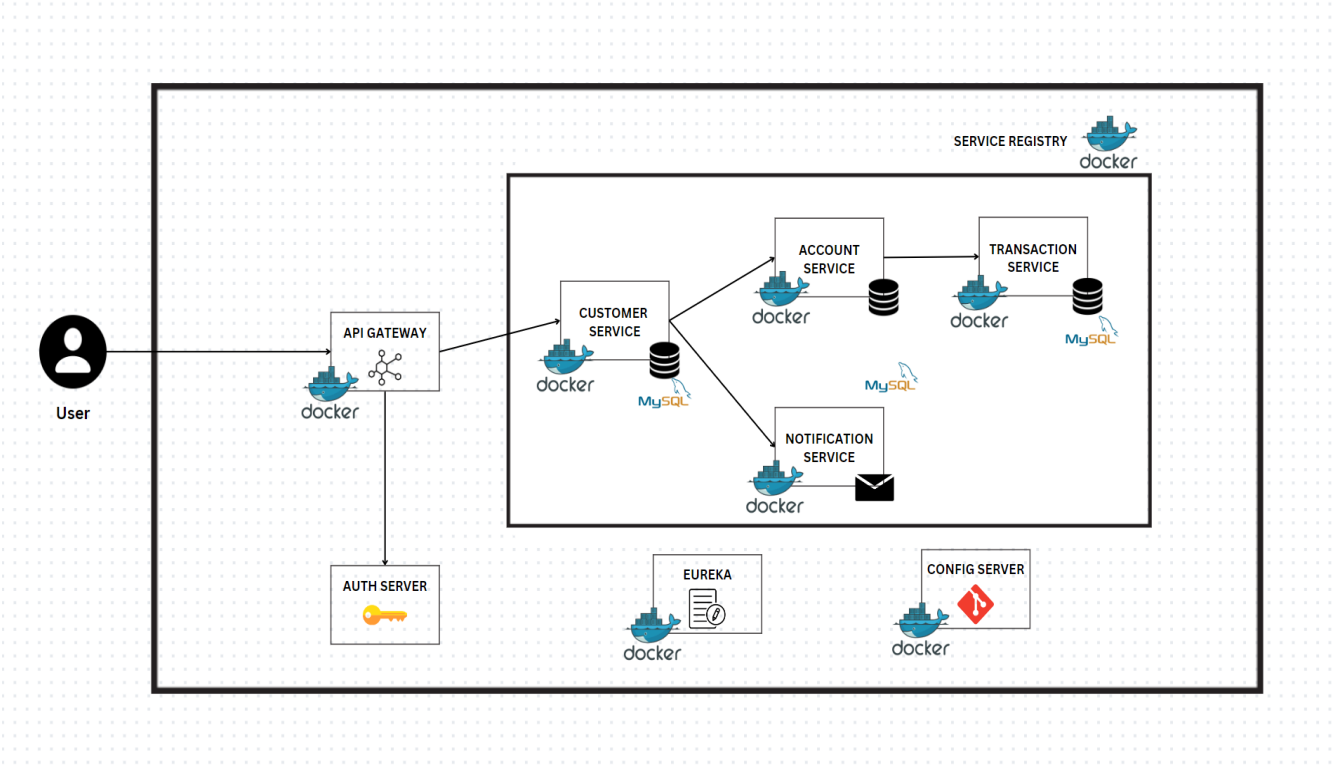
- **Backend:**
  - Java
  - Spring Boot
  - Spring Data JPA
  - Spring Security
  - Feign Client
  - OKTA OAuth2
- **Database:**
  - MySQL
- **Frontend:**
  - React
  - Node
- **Development Tools:**
  - IntelliJ IDEA IDE
  - VS Code IDE
  - Git
  - Maven
- **Deployment**:
  - Docker
  - Kubernetes

4. Architecture:

The application follows a microservices architecture. The main components include:

- **Customer Service**: Handles user registration, authentication, and profile management. Utilizes OKTA OAuth2 for secure authentication.
- **Account Service:** Manages user accounts, including creation, retrieval, and balance management. Communicates with the Customer Service for user-related operations.
- **Notification Service:** Handles email notifications. Sends custom emails for various purposes, such as transaction notifications or account updates.
- **Transaction Service:** Performs transaction-related operations, such as deposits, withdrawals, and transfers.

The microservices communicate with each other using RESTful APIs. The frontend interacts with the microservices to retrieve data and perform various operations.



5. Setup Instructions:

To set up the project locally, follow these steps:

1. Clone the project repository from GitHub.
2. Import the backend microservices and frontend project into your preferred IDE.
3. Set up the MySQL database and configure the database connection properties.
4. Set up the OKTA OAuth2 authentication service and obtain the necessary credentials.
5. Build and run each microservice using Maven or Docker.
6. Install the required dependencies for the frontend.
7. Start the frontend application.

6. Microservices:

The application is divided into the following microservices:

6.1. Customer Service:

The Customer Service is responsible for user management and authentication. It provides the following functionalities:

- User Registration: Allows users to create an account by providing necessary details.
- User Login: Authenticates users using OKTA OAuth2 and provides access to their accounts.

6.2. Account Service: The Account Service handles account-related operations. It provides the following functionalities:

- Create Account: Allows users to create a new account.
- Accounts List for Customer: Retrieves a list of accounts associated with a customer.
- Account Data for Customer: Retrieves detailed information about a specific account.

6.3. Notification Service: The Notification Service handles email notifications. It provides the following functionalities:

- Send Email on Registration: Sends an email notification to users upon successful registration.
- Send Custom Emails: Allows the system to send custom emails to users for various purposes such as transaction notifications or account updates.

6.4. Transaction Service: The Transaction Service handles transaction-related operations. It provides the following functionalities:

- Make Deposit: Allows users to deposit money into their accounts.
- Make Withdrawal: Allows users to withdraw money from their accounts.
- Transfer Money to Another Account: Enables users to transfer funds to another account.
- Show Transactions: Retrieves a list of transactions associated with a user account.

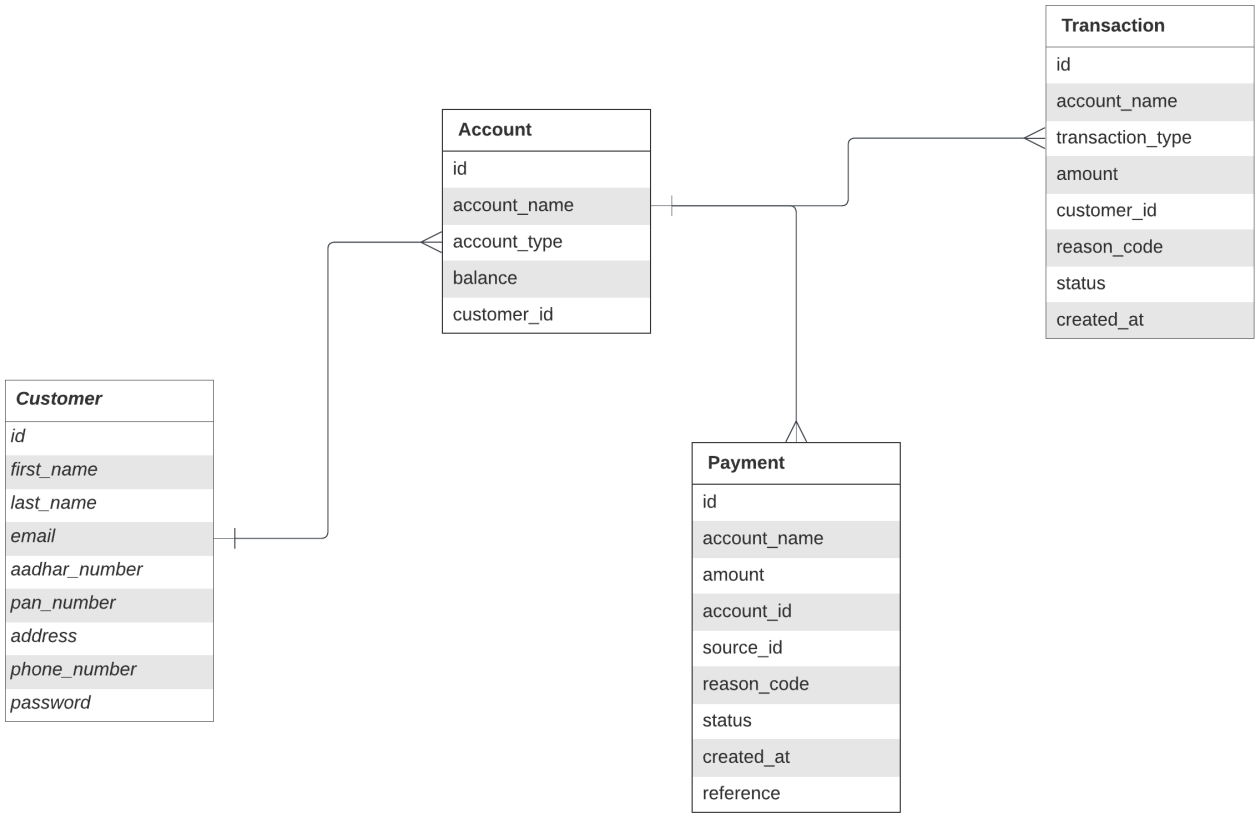
Each microservice is implemented using Spring Boot, follows the RESTful principles, and communicates with the respective databases for data storage and retrieval.

7. React Frontend:

The frontend is built using React and interacts with the microservices via RESTful APIs. It provides a user-friendly interface for users to access banking features, such as account management, balance inquiries, transaction history, and sending custom emails.

8. Database Design:

The application uses MySQL as the database management system. The database design is shown below:



Tables Schema:

Account_DB:

- id
- account_type
- balance
- customer_id
- account_name

Customer_DB:

- id
- aadhar_number
- address

- email
- first_name
- last_name
- pan_number
- phone_number
- password

Transaction_DB:

- id
- account_name
- amount
- status
- transaction_type
- created_at
- reason_code
- customer_id

9. Security:

Spring Security is utilized for authentication and authorization, with OKTA OAuth2 integration for secure authentication.

10. Testing:

The project includes unit tests and integration tests to ensure the reliability and correctness of the system. Backend tests use JUnit and Mockito, while frontend tests use React Testing Library.

11. Deployment:

The application can be deployed using Docker, which allows containerization of each microservice and simplifies the deployment process.

12. Conclusion:

The Spring Boot Microservices and React-based Banking Application provides a robust and scalable banking system. It leverages microservices architecture to enhance modularity, scalability, and maintainability. The combination of Spring Boot, React, and other

technologies ensures secure authentication, efficient data management, and a user-friendly interface.

13. References:

- Spring Boot Documentation
- React Documentation
- MySQL Documentation
- OKTA OAuth2 Documentation
- Docker Documentation