**Global Bank Management Application**

**Description:**

The main objective of this application is to provide services of Global Bank to its customers to have banking-related services through a web application and performs simple bank transactions. This application will allow performing the below operations, here there are two types of users are there one is a customer who has an account in their branch and another admin( usually the bank internal employee who will manage the data and functionalities of the application).

The application is suppose to create microservices based architecture, where two different services such as

1. User-service
2. Admin-service

Where user-service is expected to perform the below said operations:

1. A user he can register him/her selves to the application
2. As a user, he/she can be able to apply for a loan in a specific branch.
3. As a user he/she can be able to perform withdrawal or deposit amount
4. As a user can be able to view all his / her selected transactions ie., withdraw or deposit for a specific period.
5. Display all transactions for a specific period.

Where Admin-service is expected to perform the below said operations:

1. Admin can be able to delete any customer
2. Admin can be able to add branch details
3. Admin can be able to delete branch details
4. Admin can be able to open an account for registered customer
5. Admin can be able to close an account for registered customers
6. Display all customers details
7. Displays all branch related details
8. See all the loans of customers

Every service will expose their API’s publicly so that other services and be able to call them by using REST.

**3. Toolchain**

Databases: MySQL

Backend processing: Spring and Springboot

Database Connectivity: Spring data JPA

Version control systems: Git

Build Tools: Maven

**Business Requirement:**

There will be 5 main user stories required to be implemented covering the below use cases:

**User Stories**

Note: A separate service component must be created to call the spring boot backend services and all the validations or processing regarding the use case should be done at the backend only.

**Backend Layer User Stories for User-Service**

|  |  |
| --- | --- |
| User story Id | Us-01 |
| User story title | User Login processing |
| User Story Details | 1. Should be able to extract the values from request body using @RequestBody 2. Read the user details from database using spring data jpa and validate it with the UI values. 3. After validating should send response 4. Must use GET Method of communication |
| Acceptance Criteria | 1. Once user validation is done, view must return true or false 2. If validation fails, view must display appropriate messages or exceptions 3. All validations must be performed at backend only |

|  |  |
| --- | --- |
| User story Id | Us-02 |
| User story title | User Registration processing |
| User Story Details | 1. Should be able to extract the values from request body object 2. Read the user registration values from UI and pass it to service layer further to dao layer to perform insertion of record in database 3. Should return the model object after successful insertion of data 4. Must use POST method of communication |
| Acceptance Criteria | 1. Validation must be performed for all the user registration data 2. If validation fails, throw appropriate custom exception |

|  |  |
| --- | --- |
| User story Id | Us-03 |
| User story title | Loan Application |
| User Story Details | 1. Should be able to extract the values from request body object 2. Read the loan application details from UI and pass it to service layer further to dao layer to perform insertion of record in database 3. Should return the Boolean object after successful loan submission 4. Must use POST method of communication |
| Acceptance Criteria | 1. Loan application can be accepted from the account holders only |

|  |  |
| --- | --- |
| User story Id | Us-04 |
| User story title | Transaction Management |
| User Story Details | 1. Should be able to read transactional ie user account\_number, transaction\_type and period details from UI 2. Search from the transactional\_details database based on the values received 3. Should return all transaction data |
| Acceptance Criteria | 1. Transaction type must be either withdraw or deposit |

**Backend layer user stories for Admin-Service**

|  |  |
| --- | --- |
| User story Id | Us-01 |
| User story title | Admin Login |
| User Story Details | 1. Should be able to extract the values from request body using @RequestBody 2. Read the user details from database using spring data jpa and validate it with the UI values. 3. After validating should send Boolean value true if validated else false 4. Must use GET Method of communication |
| Acceptance Criteria | 1. If validation fails or no password or user id provided raise custom exception 2. All validations must be performed at backend only |

|  |  |
| --- | --- |
| User story Id | Us-02 |
| User story title | Bank branch registration |
| User Story Details | 1. Should be able to extract the values from request body object 2. Read the user registration values from UI and pass it to service layer further to dao layer to perform insertion of record in database 3. Should return the model object after successful insertion of data 4. Must use POST method of communication |
| Acceptance Criteria | 1. Once branch details are successfully inserted, must return Boolean value 2. If failed appropriate exception should be raised and handled. |

|  |  |
| --- | --- |
| User story Id | Us-03 |
| User story title | Customer Account Creation |
| User Story Details | 1. Should be able to extract the values from request body object 2. Read the customer account details UI and pass it to service layer further to dao layer to perform insertion of record in database 3. Should return the Boolean object after successful customer account submission 4. Must use POST method of communication |
| Acceptance Criteria | 1. Only registered customers can open an account. 2. Account type can either savings or current |

|  |  |
| --- | --- |
| User story Id | Us-04 |
| User story title | Display Customer Details |
| User Story Details | 1. Should fetch all the customer details from the database and return it in the form of list. |
| Acceptance Criteria | 1. Should return only in the java list object |

|  |  |
| --- | --- |
| User story Id | Us-05 |
| User story title | Display Branch Details |
| User Story Details | Should fetch all the branch details from the database and return it in the form of list. |
| Acceptance Criteria | Should return only in the java list object |

**Note: All responses should send using ResponseEntity by embedding appropriate data and response status also exceptions must be handled through custom ie., user-defined exceptions only.**

**Architecture:**

1. Develop two different services running in two different servers.
2. Each service must be registered and communicated through Service Discovery server only.
3. All services must be accessed through a gateway only, all user or admin validations should be implemented in the gateway only.
4. Fault tolerance or resiliency must be handled in the services using resiliency4j or Hystrix.
5. A cloud config server must be used to read the configurations for each of the service