Banking application:

You will need the following classes:

- **Class Customer(custId, cust-name, address)**
- **Class CustAccount(acctId, custId, bal)**

You will need the following microservices:

customer-details-service : provides following endpoints

- /customers/id : GET : get customer with id
- /customers/ : POST : create new customer

Account-details-service: provides following endpoints

- /accounts/custid : GET : get account details for this cust-id

Bank-manager-service: provides following endpoints

- /accounts/custid : GET : Given custid, must retrieve account details for that customer from Account-details-service as well as customer details from customer-details-service.

Create an Angular application that has following functionalities:

- Create a BankService that will access the microservices via the HttpClientModule.
- Accept customer details in HTML form. The component will use the Bankservice that will use customer-details-service endpoints to save customer details into repository
- Accept cust-id in HTML form. The component will use the Bankservice that will use Bank-manager-service endpoints to access customer details as well as his account details. Display the retrieved information in template as a neat table

Note:

- For the repository at the backend (at microservices end), create DAO, preferably use database in spring boot. Only If time does not permit, then use simple collection in DAO
- Register all microservices on Eureka server. Use RestTemplate to enable intercommunication in microservices
- Use very simple template-driven forms for the application. No need for validation etc