Problem Statement

In the session, you will be working on developing an end-to-end application that makes use of multiple Spring Cloud projects.

The agenda of this exercise is for you to realize how you can make use of the Spring Cloud projects that you have learnt as part of this module for a real-world application.

This will be called the '**Cast my vote**' application.

The **functionality** of the application is described below:

The application will have two types of user-profiles: User and Candidate.

Only a regular user will be able to register on the application to cast their vote. Candidate details are pre-populated in the database and shouldn't be touched.

Each user can register themselves through the user registration API.

Each user can cast their vote basis the condition that they are above the age of 18.

Users will be able to see the list of candidates at the time of casting the vote.

Counting microservice can see the results of the voting in real-time. As soon as a vote is cast, the counting microservice will be able to see the updated count of the votes.

The **Spring Cloud projects** that can be used in the application are:

- Spring Cloud Eureka (service registry and discovery)
- Spring Cloud Gateway (API gateway)
- Spring Cloud Function (<to verify the age of the user for casting vote>)
- Spring Cloud Openfeign (<to fetch user details and candidate details from different microservices>)
- Spring Cloud Stream (for Admin to subscribe to topics to get real-time counting for votes).

The current structure of the application is as follows:

- The application contains x number of microservices.
    - User Microservice
    - Voting Microservice
    - Counting Microservice
    - Eureka Server
    - API Gateway
- APIs included in each microservice is as follows:
    - User microservice:
        - Register a User
        - Get list of candidates
        - Get user detail from id
    - Voting microservice:
        - Cast a Vote
- Communication channels between microservices are as follows:
    - Voting microservice synchronously calls User microservice to fetch user details.
    - Voting ms synchronously calls User microservice to fetch and verify candidate details.
    - Admin has subscribed to **VotingBoard** topic to fetch the real-time vote count for this topic in Kafka.

TODOS:

The following parts of the stub code need to be complete to successfully run the application.

1. Create the following kafka topics :

   Microservice: Voting Microservice → Topic: VotingBoard

   Microservice: Counting Microservice → Topic: VotingBoard

2. Add the following dependencies in pom.xml

   Microservice: User Microservice
   Dependencies: Spring Boot web, Spring Data JPA, H2, Lombok, ModelMapper, Spring Cloud Starter Function, Eureka Client

Microservice: Voting Microservice
Dependencies: Spring Boot web, Lombok, ModelMapper, Spring cloud Starter Feign, Spring cloud Stream, Spring cloud starter stream kafka, Eureka Client

Microservice: Counting Microservice
Dependencies: Spring Boot Web, Spring Cloud stream, Spring cloud Stream Binder Kafka, Eureka Client

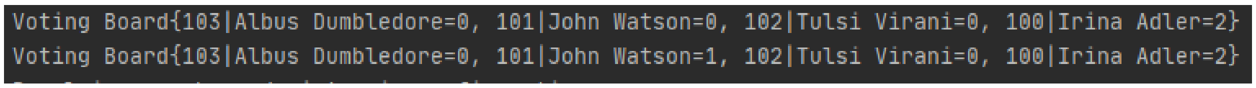
3. Add the following annotations/properties wherever appropriate as per your understanding of these annotations or properties
    a. @EnableEurekaServer
    b. spring.cloud.function.scan.packages
    c. @EnableFeignClients
    d. @EnableBindings
    e. Defining eureka endpoints in application.yml
    f. Define wild card in API Gateway
    g. @LoadBalanced
    h. Define Kafka topics in application.yml
    i. @StreamListener

4. Add functionality for the Counting microservice to subscribe to the Kafka topics.

    Microservice: Counting Microservice
    Topic : VotingBoard

    TODO 1: complete the method <consumeMessage>

    Expected Request:

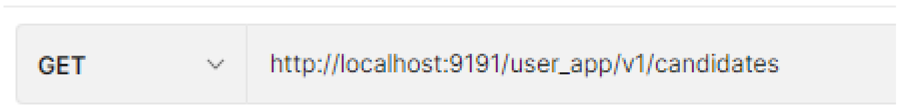


```
Voting Board{103|Albus Dumbledore=0, 101|John Watson=0, 102|Tulsi Virani=0, 100|Irina Adler=2}
Voting Board{103|Albus Dumbledore=0, 101|John Watson=1, 102|Tulsi Virani=0, 100|Irina Adler=2}
```

5. Add functionality for User Microservice:

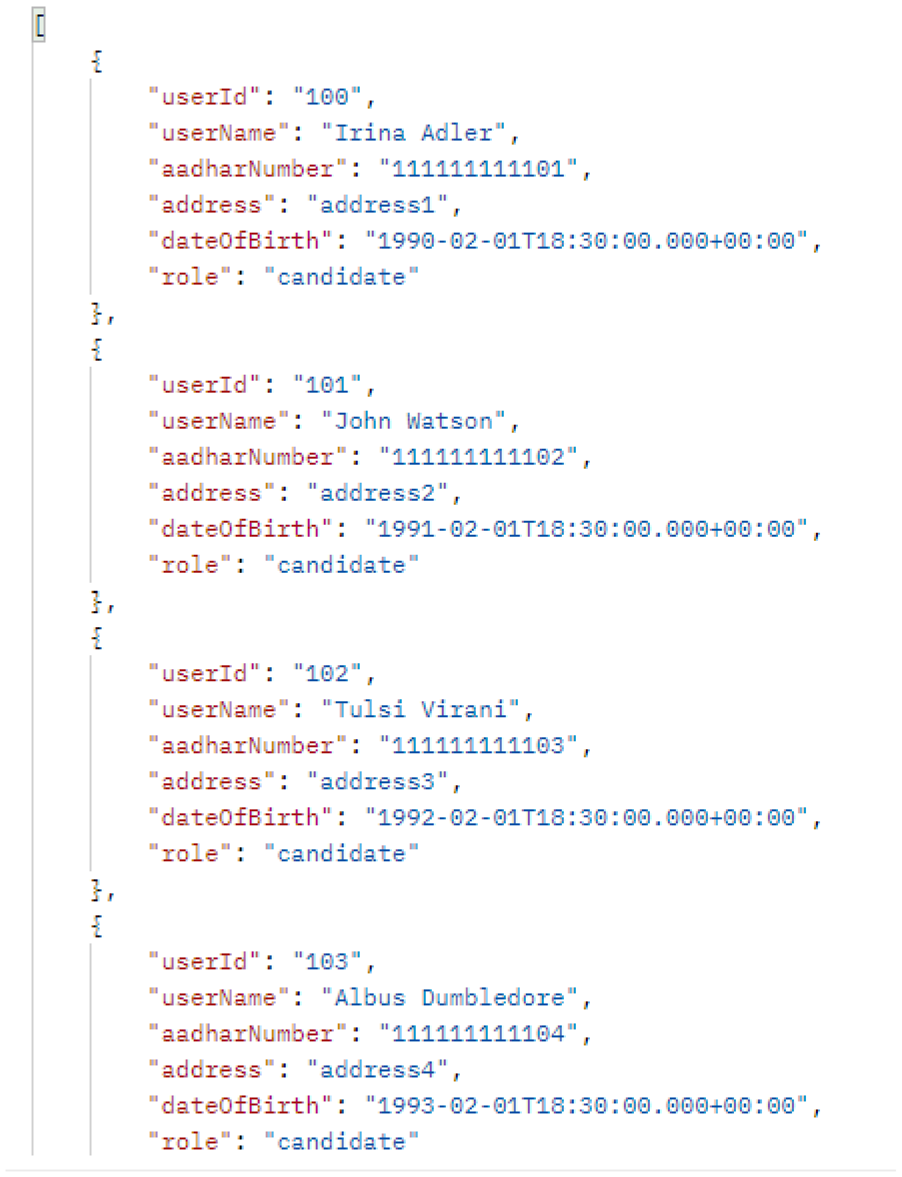
    TODO 1: complete the method to get the list of candidates

    Request:



```
GET    http://localhost:9191/user_app/v1/candidates
```

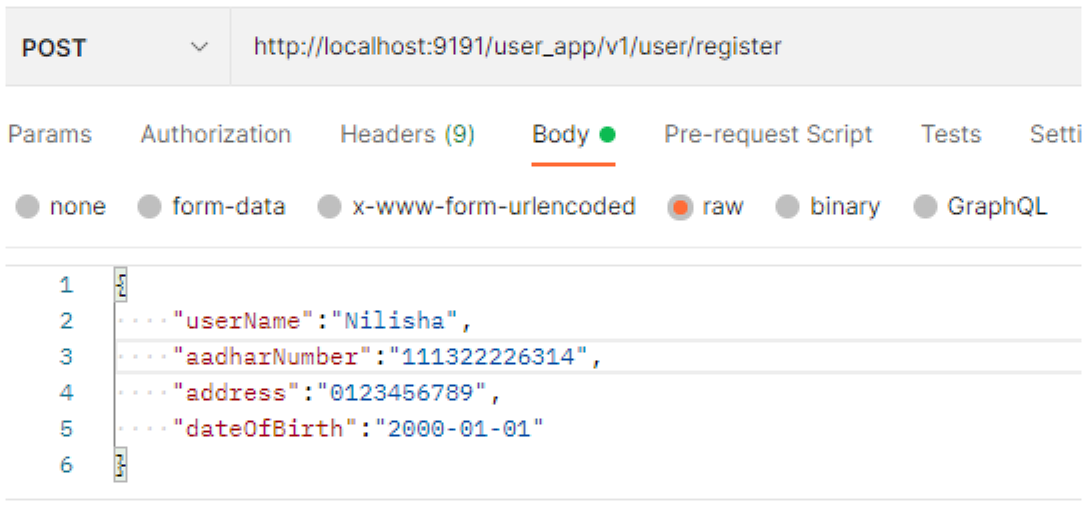
Response :



```
[
    {
        "userId": "100",
        "userName": "Irina Adler",
        "aadharNumber": "111111111101",
        "address": "address1",
        "dateOfBirth": "1990-02-01T18:30:00.000+00:00",
        "role": "candidate"
    },
    {
        "userId": "101",
        "userName": "John Watson",
        "aadharNumber": "111111111102",
        "address": "address2",
        "dateOfBirth": "1991-02-01T18:30:00.000+00:00",
        "role": "candidate"
    },
    {
        "userId": "102",
        "userName": "Tulsi Virani",
        "aadharNumber": "111111111103",
        "address": "address3",
        "dateOfBirth": "1992-02-01T18:30:00.000+00:00",
        "role": "candidate"
    },
    {
        "userId": "103",
        "userName": "Albus Dumbledore",
        "aadharNumber": "111111111104",
        "address": "address4",
        "dateOfBirth": "1993-02-01T18:30:00.000+00:00",
        "role": "candidate"
```

TODO 2:

Request : Register a user



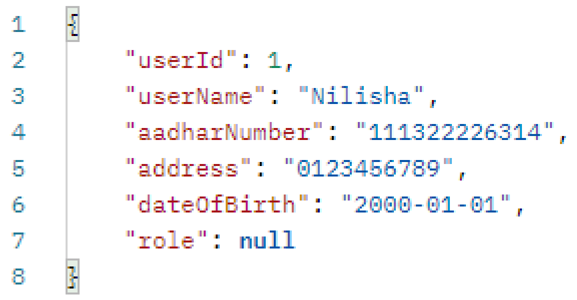
```
POST    http://localhost:9191/user_app/v1/user/register
```

Params   Authorization   Headers (9)   Body ●   Pre-request Script   Tests   Setti

○ none   ○ form-data   ○ x-www-form-urlencoded   ● raw   ○ binary   ○ GraphQL

```
{
    "userName":"Nilisha",
    "aadharNumber":"111322226314",
    "address":"0123456789",
    "dateOfBirth":"2000-01-01"
}
```

Response :

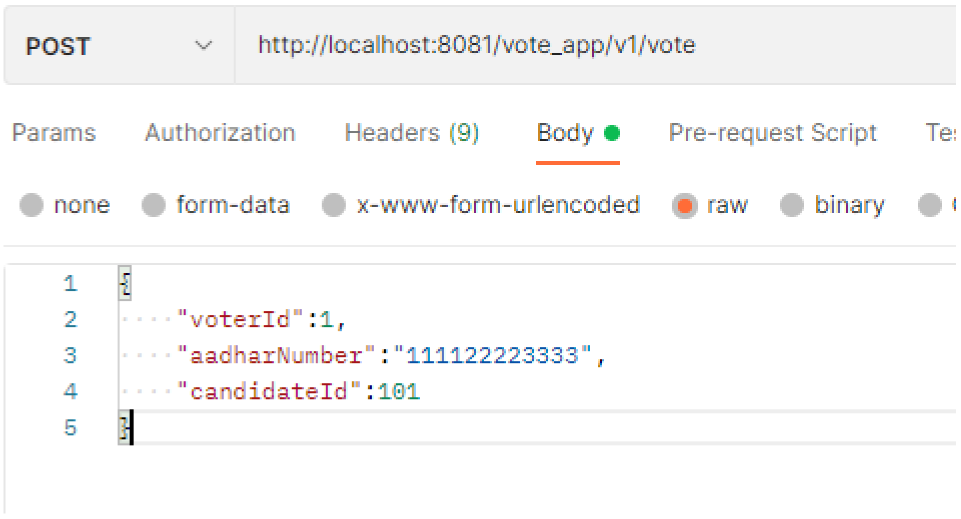


```
{
    "userId": 1,
    "userName": "Nilisha",
    "aadharNumber": "111322226314",
    "address": "0123456789",
    "dateOfBirth": "2000-01-01",
    "role": null
}
```

6. Add functionality for Voting Microservice

TODO 1: Implement Cast vote functionality

Request :



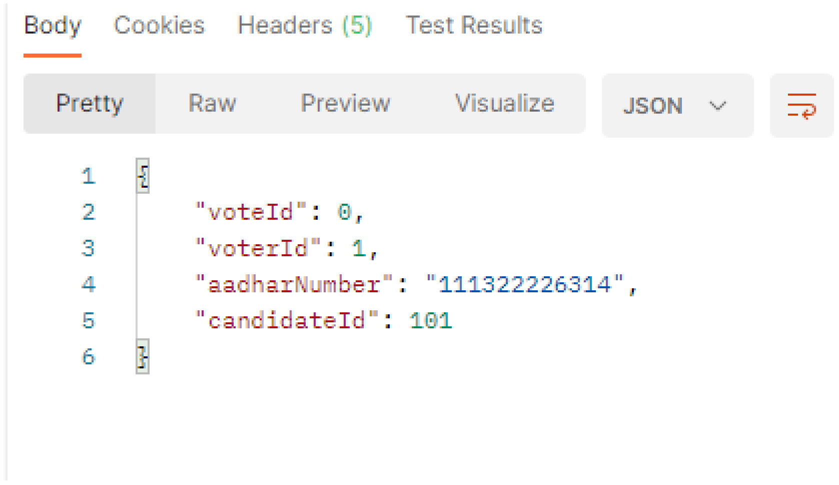
POST http://localhost:8081/vote_app/v1/vote

Params | Authorization | Headers (9) | Body ● | Pre-request Script

none | form-data | x-www-form-urlencoded | [✓ raw] | binary

```
{
    "voterId":1,
    "aadharNumber":"111122223333",
    "candidateId":101
}
```

Response :

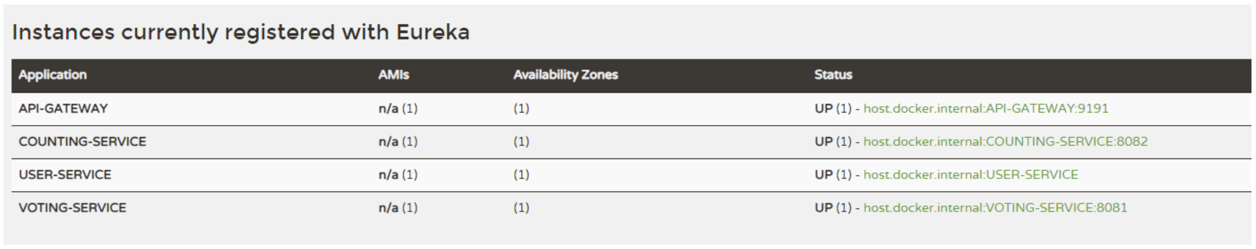


Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
{
    "voteId": 0,
    "voterId": 1,
    "aadharNumber": "111322226314",
    "candidateId": 101
}
```

Make sure that all microservices are getting registered in Eureka Server



Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - host.docker.internal:API-GATEWAY:9191
COUNTING-SERVICE	n/a (1)	(1)	UP (1) - host.docker.internal:COUNTING-SERVICE:8082
USER-SERVICE	n/a (1)	(1)	UP (1) - host.docker.internal:USER-SERVICE
VOTING-SERVICE	n/a (1)	(1)	UP (1) - host.docker.internal:VOTING-SERVICE:8081