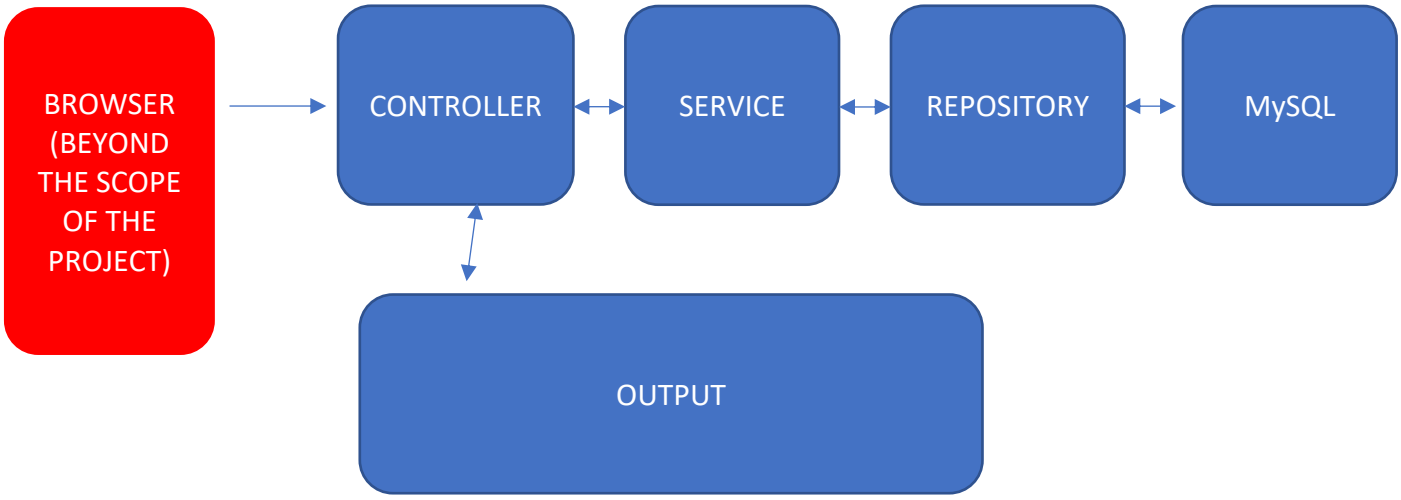
DESIGN DOCUMENT

GITHUB REPOSITORY: https://github.com/shreenath96/hr

SYSTEM-DESIGN:

Components:

- **User:**
  - UserController
  - User model
  - UserService
  - UserServiceImpl
  - User Repository
- **UserAddress:**
  - UserAddress Controller
  - UserAddress model
  - UserAddress Service
  - UserAddress ServiceImpl
  - UserAddress Repository



Notes:

- The controller components contain all the GET/POST/PUT/DELETE methods.
- The model component contains the data fields of the User and UserAddress Tables.
- The repository of both User and UserAddress extends JpaRepository.
- The Service component contains the methods to save,view,find etc methods that the controller uses.
- The ServiceImpl component contains all the implementation of the above-mentioned methods.
- Enums have been created for Gender, Usertype and Addresstype.

DATABASE STRUCTRE:
SCHEMA NAME: EMPLOYEE
ENTITIES:

- **User:**

  Attributes:

annualSalary	**number**
dateOfBirth	**string** ***example: yyyy-MM-dd***
email	**string**
firstName	**string**
fullName	**string**
gender	**stringEnum:** **[FEMALE, MALE, OTHER]**
id	**integer($int64) ----------(primary key)**
lastName	**string**
mobilePhone	**string**
userType	**stringEnum:** **[CONSULTANT, EMPLOYEE]**

- **UserAddress:**

  Attributes:

addrLn1	-	**string**
addrLn2	-	**string**
addrName	-	**string**
addrType	-	**stringEnum:** **[BILLING, MAIN, SHIPPING]**
city	-	**string**
country	-	**string**
id	-	**integer($int64)------- -(primary key)**
locationCode	-	**string**
postalCode	-	**string**
stateCode	-	**string**
userId	-	**integer($int64)------- -(foreign key)**

STEPS ON TESTING THE APPLICATION/API:

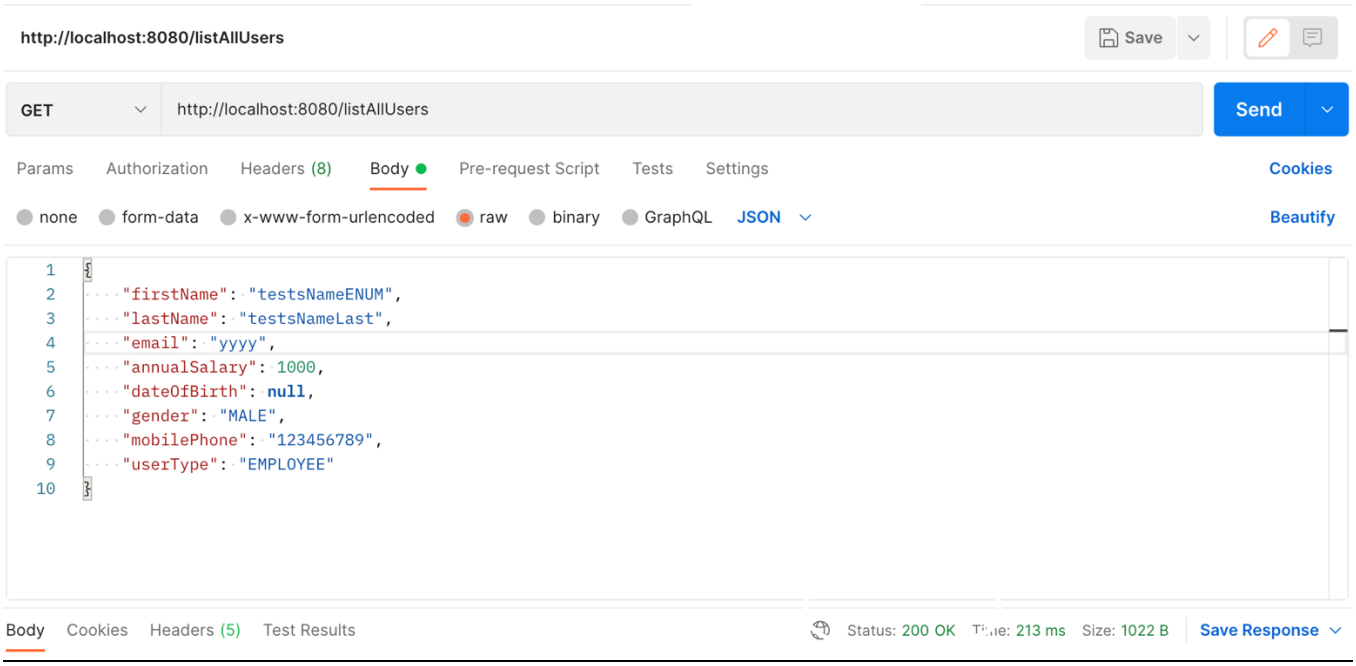
STEP:1
Locate the API tag and method type from the respective Controller.

STEP:2
- Open any API testing suite like “Postman”.

STEP3:
- Workspace -> Collections
- Connect to local host with the code: http://localhost:8080/<APItag> note:replace APItag
- Click Send

STEP4:
- Status 200 OK indicate the API was tested successfully.
- Any other status message indicates an error.

PROOF OF API’s working:



http://localhost:8080/usersAddress

Save

POST http://localhost:8080/usersAddress Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookie

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautif

```
{
    "firstName": "testsNameENUM",
    "lastName": "testsNameLast",
    "email": "yyyy",
    "annualSalary": 1000,
    "dateOfBirth": null,
    "gender": "MALE",
    "mobilePhone": "123456789",
    "userType": "EMPLOYEE"
}
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 91 ms Size: 182 B Save Response

http://localhost:8080/listAllUsers

Save

GET http://localhost:8080/listAllUsers Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "firstName": "testsNameENUM",
    "lastName": "testsNameLast",
    "email": "yyyy",
    "annualSalary": 1000,
    "dateOfBirth": null,
    "gender": "MALE",
    "mobilePhone": "123456789",
    "userType": "EMPLOYEE"
}
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 28 ms Size: 1022 B Save Response

http://localhost:8080/users

Save

POST http://localhost:8080/users Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "firstName": "testsNameENUM",
    "lastName": "testsNameLast",
    "email": "yyyy",
    "annualSalary": 1000,
    "dateOfBirth": null,
    "gender": "MALE",
    "mobilePhone": "123456789",
    "userType": "EMPLOYEE"
}
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 27 ms Size: 182 B Save Response

http://localhost:8080/users/3

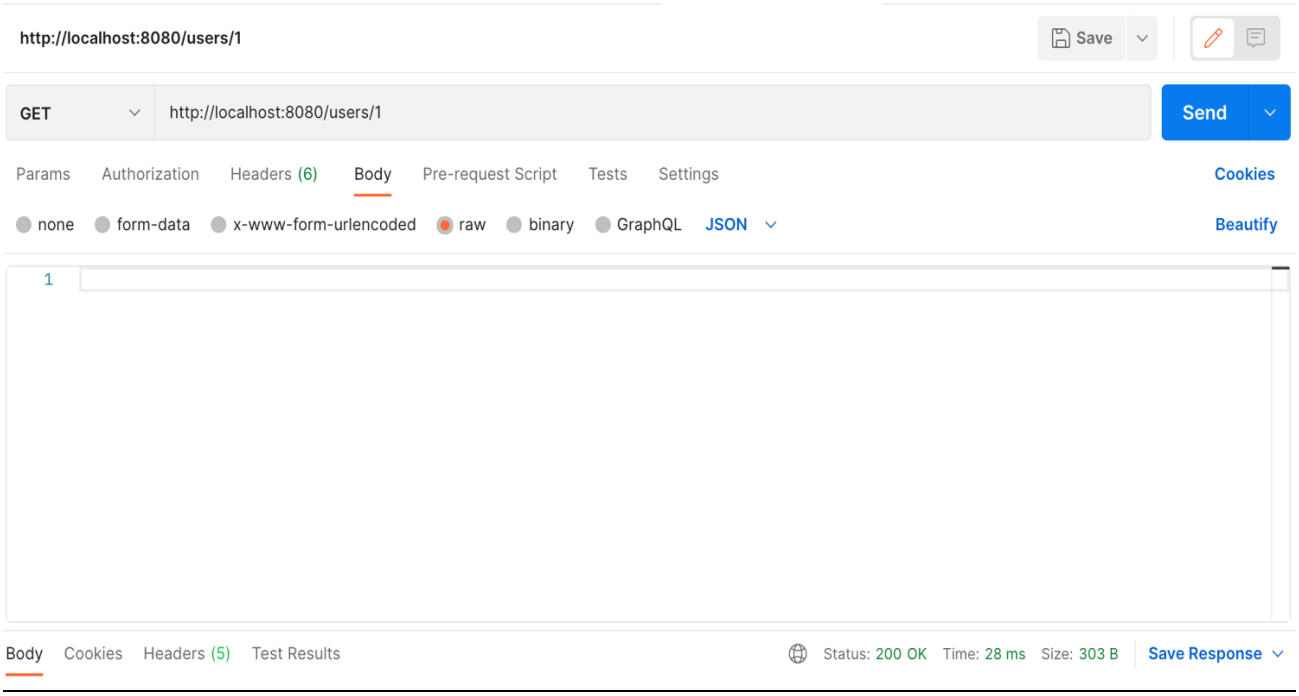
Save

DELETE http://localhost:8080/users/3 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

Body Cookies Headers (5) Test Results Status: 200 OK Time: 33 ms Size: 184 B Save Response



CONSTRAINTS WITH THE APPLICATION:

- I have not yet implemented the User Summary- GET method due to time constraint.
- The application doesn't include UI/Front-End components.
- The application only includes the backend part developed using SPRINGBOOT(IDE), MySQL (database), Spring Data JPA, Hibernate.
- Possible improvements to the system include : adding html pages with css stylesheets for better user interface, complex data structure such as hashmap to store multiple addresses for a user etc.