Documentation

Manasvi Nitin Bhojak

8329014552

manasvibhojak@gmail.com

Assignment 2: Java Backend

Create Spring boot application

Spring Boot provides a web tool called **Spring Initializer** to bootstrap an application quickly. **https://start.spring.io/** and generate a new spring boot project.

I have use the below details in the Spring boot creation:

Project Name: SocialMediaApp

Project Type: Maven

Choose dependencies: Spring Web, Spring Data JPA, Spring Security, Dev Tools and MySQL Driver

Package name: com.app

Packaging: war

Configure MySQL Database

Since I am using MySQL as our database, we need to configure the database **URL**, **username**, and **password** so that Spring can establish a connection the database on startup. Open src/main/resources/application.properties file

```
server.port=8080
server.tomcat.uri-encoding=utf-8

#database configuration
spring.datasource.url=jdbc:mysql://localhost:3306/socialmediaapp?serverTimezone=UTC&useUnicode=y
es&characterEncoding=UTF-8&rewriteBatchedStatements=true
spring.datasource.username=root
spring.datasource.password=Mysql@0306
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Model Layer - Create JPA Entities

1stly I have create *User* JPA entities with the variables like userId which is @Id and @GeneratedValue(strategy = GenerationType.IDENTITY) and the username with unique=true and password which is not null and the email And I have created the setter and getter on it . after that perform some operation through database I have created repository layer

Repository Layer

The repository contains the various methods for getting the records from the database. It is also used to create a user-defined method to get records/data from the database.

```java
@Repository
public interface UserRepository extends JpaRepository<User,Integer> {

	 public User findByUsername(String username);

	public Boolean existsByUsername(String username);

	public Boolean existsByEmail(String email);

	public Boolean existsByPassword(String password);

	public User findByUserId(int userId);

}
```

Service implementation Layer

The service is used to perform various operations (e.g. CRUD) on the database. The service file is used to write business logic in the code.

```
@Service

public class UserServiceImpl implements UserService {

        @Autowired

        private UserRepository userRepo;

        @Override

        public User registerUser(User user) {

                User save = userRepo.save(user);

                return save;

        }

        public Boolean existsByPassword(String password) {

                Boolean existsByPassword = userRepo.existsByPassword(password);

                return existsByPassword;

        }

        public Boolean existsByUsername(String username) {

                Boolean existsByUsername = userRepo.existsByUsername(username);

                return existsByUsername;

        }

        @Override

        public User findByUserId(int userId) {

                User findByUserId = userRepo.findByUserId(userId);

                return findByUserId;

        }

        @Override

        public Boolean existsByEmail(String email) {

                Boolean existsByEmail = userRepo.existsByEmail(email);

                return existsByEmail;

        }

}
```

1. Create Account

A controller is a very important part of Spring Boot MVC. It is used to create API endpoints and interact with users via request and response. It uses the Service that we have created previously to perform the various operation to manipulate the data so that we get all data in the proper format.

```
@PostMapping("/Register")
public ResponseEntity<?> RegisterUser(@RequestBody User user) {
            User registerUser = new User();
            String      regex       ="^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^-]+(?:\\\\.[a-zA-Z0-
9_!#$%&'*+/=?`{|}~^-]+)*@[a-zA-Z0-9-]+(?:\\\\.[a-zA-Z0-9-]+)*$";

      Pattern pattern = Pattern.compile(regex);
      String email = user.getEmail();

      if(userService.existsByUsername(user.getUsername())){
            return new ResponseEntity<>("Username is not available please try something
else", HttpStatus.BAD_REQUEST);
             }

            if(userService.existsByEmail(user.getEmail())){
                   return new ResponseEntity<>("email is already used please use other
email other wise login with this email", HttpStatus.BAD_REQUEST);
             }

       if(!pattern.matcher(email).matches()) {
                   return  new  ResponseEntity<>("please  enter  valid  email  address",
HttpStatus.BAD_REQUEST);
            }


            //registration after all validation
            registerUser.setUsername(user.getUsername());
            registerUser.setPassword(passwordEncoder.encode(user.getPassword()));
            registerUser.setEmail(user.getEmail());
            registerUser.setRoles("ROLE_NORMAL");
            User registeredUser = userService.registerUser(registerUser);

            return ResponseEntity.ok(registeredUser);
      }
```

CustomUserDetail

```
public class CustomUserDetail implements UserDetails {
	private static final long serialVersionUID = 1L;
	@Autowired
	private User user;
	public CustomUserDetail(User user) {
		this.user = user;
	}

	@Override
	public Collection<? extends GrantedAuthority> getAuthorities() {
		HashSet<SimpleGrantedAuthority> set = new HashSet<>();
		set.add(new SimpleGrantedAuthority(this.user.getRoles()));
		return set;
	}
	@Override
	public String getPassword() {
		return this.user.getPassword();
	}
	@Override
	public String getUsername() {
		return this.user.getUsername();
	}

	@Override
	public boolean isAccountNonExpired() {
		return true;
	}

	@Override
	public boolean isAccountNonLocked() {
		return true;
	}

	@Override
	public boolean isCredentialsNonExpired() {
		return true;
	}

	@Override
	public boolean isEnabled() {
		return true;
	}

}
```

CustomUserDetailsService

I have write a logic to load user details by username from the database.
I have create a ***CustomUserDetailsService*** which implements ***UserDetailsService*** interface (Spring security in-build interface) and provides a implementation for ***loadUserByUername()*** method:

```
@Service
public class CustomUserDetailsService implements UserDetailsService {

	@Autowired
	private UserRepository userRepository;

	@Override
	public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
		User user = userRepository.findByUsername(username);
		return new CustomUserDetail(user);
	}

}
```

Spring Security uses the **UserDetailsService** interface, which contains the **loadUserByUsername(String username)** method to lookup **UserDetails** for a given username. The **UserDetails** interface represents an authenticated user object and Spring Security provides an out-of-the-box implementation of **org.springframework.security.core.userdetails.User.**

Spring Security Configuration

I have created a class ***SecurityConfig*** and added the following configuration to it:

SecurityConfig class extends ***WebSecurityConfigurerAdapter*** and overrides some of its methods to provide custom security configurations.

The ***@EnableWebSecurity*** is the primary spring security annotation that is used to enable web security in a project.

The ***@EnableGlobalMethodSecurity*** is used to enable method-level security based on annotations.

```
@Configuration

@EnableWebSecurity

public class MySecurityConfig  extends WebSecurityConfigurerAdapter{

        @Autowired

        private CustomUserDetailsService customUserDetailService;

        @Override

        protected void configure(HttpSecurity http) throws Exception {

                http.csrf().disable().authorizeRequests() .antMatchers("/user/**").permitAll()

                 .antMatchers("/post/**").permitAll() .anyRequest().authenticated() .and().httpBasic();

        }

        @Override

        protected void configure(AuthenticationManagerBuilder auth) throws Exception {

                auth.userDetailsService(customUserDetailService).passwordEncoder(passwordEncoder());

        }

        @Bean

        public AuthenticationManager authenticationManagerBean() throws Exception {

            return super.authenticationManagerBean();

         }

        @Bean

        public PasswordEncoder passwordEncoder() {   return new BCryptPasswordEncoder();

                }
```

2. Login

```
@PostMapping("/signin")
public ResponseEntity<String> authenticateUser(@RequestBody User user){
//check username exit or not
if(!userService.existsByUsername(user.getUsername())){
      return new ResponseEntity<>("Username is not find!", HttpStatus.BAD_REQUEST);
}

//after checking username and password authentication done
Authentication authentication = authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(user.getUsername(), user.getPassword()));

             SecurityContextHolder.getContext().setAuthentication(authentication);
                    return new ResponseEntity<>("User signed-in successfully!.",
HttpStatus.OK);
}
```

uploading a file 1st I have to set some properties in the application.properties file

```
spring.servlet.multipart.enabled=true
spring.servlet.multipart.file-size-threshold=2KB
spring.servlet.multipart.max-file-size=200MB
spring.servlet.multipart.max-request-size=215MB
```

and then to upload a file to the target location I have to create Filehendler class. In that I have write a uploadfile method.

```
public boolean uploadFile(MultipartFile multipartFile)
       {
             boolean f=false;
             try
             {
      Files.copy(multipartFile.getInputStream(),Paths.get(UPLOAD_DIR+File.separator+multipartFile.getOriginalFilename()),StandardCopyOption.REPLACE_EXISTING);
                   f=true;

             } catch (Exception e) {
                        e.printStackTrace();
             }
             return f;
}
```

```
public class Post {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Integer id;

@ManyToOne
@JoinColumn(name = "userId")
private User user;

private String postImg;

@Column(name="createdDate", columnDefinition="TIMESTAMP DEFAULT CURRENT_TIMESTAMP")
    private Date createdDate;
//getter and setter
```

In this entity user and post have many to one relationship because one user can upload the many post for that I have use the @ManyToOne annotation and @JoinColumn annotation

In the repository level I need one custom(user-defined) method to get the user-specific posts. All other methods (e.g findByUser) will be provided by default as we extend the JpaRepository interface.

Repository layer of Post.

```
public interface PostRepository extends JpaRepository<Post, Integer> {

            public List<Post> findByUser(User user);

}
```

Service implantation layer of Post.

```
@Autowired
private PostRepository postRepository;

@Override
public List<Post> findByUser(User user) {
            List<Post> findByUser = postRepository.findByUser(user);
            return findByUser;
}

@Override
public Post addPost(Post post) {
            Post save = postRepository.save(post);
            return save;
}
```

```
@PostMapping("/{userId}/upload-post")
public      ResponseEntity<?>      uploadFile(@RequestParam("file")      MultipartFile
file,@PathVariable("userId") int userId)
{
      try
             {
                   if(file.isEmpty())
                   {
                         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Request   must   contain
file");
                   }

                   boolean f= fileUploadHelper.uploadFile(file);
                   if(f)
                   {
                  Date utilDate = new Date();
                  java.sql.Date date = new java.sql.Date(utilDate.getTime());
                  User user = userService.findByUserId(userId);
                  Post p = new Post();
                  p.setUser(user);
                  p.setPostImg(ServletUriComponentsBuilder.fromCurrentContextPath()
                  .path("/image/").path(file.getOriginalFilename()).toUriString());
                         p.setCreatedDate(date);

                         Post save = postService.addPost(p);
                         return ResponseEntity.ok(save);

                   }

             }catch (Exception e) {
                  e.printStackTrace();
            }
             //if some othe error occure
             return                                     ResponseEntity.status(HttpStatus.
INTERNAL_SERVER_ERROR).body("Some thing went wrong time again leter");
       }
```

4.Get all images uploaded by the current user

in this method I am using a FindByUser method to getting the all images that uploaded by current user.

```
@GetMapping("allPost/{userId}")
       public ResponseEntity<?> getPostByUser(@PathVariable("userId") int userId) {

             //display the post of perticular user
             List<Post> list =
postService.findByUser(userService.findByUserId(userId));

             //check the post is there or not
             if(list.isEmpty()) {
                   return ResponseEntity.status(HttpStatus.NOT_FOUND).body("NO POST");
            }

             //to display all the post
             return ResponseEntity.ok(list);
}
```

5.Show a list of all followers and following

For this I have create a FollowData JPA entity. In FollowData model fid is for unique identity and userId is for to know which user have that follower and User is for follower information and that have ManyToOne relationship because one user can follow many followers. And the following entity is for to know that user is following that particular follower or not if following value is 1 then it indicates user follow that follower if following value is 0 than it indicates user is not follow that follower

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int fid;

private int userId;

@ManyToOne
@JoinColumn(name = "follwer_id")
private User user;

private int following;
```

Repository layer of FollowData.

```
public List<FollowData> findByUserId(int uid);

@Query("From FollowData f  where f.userId = ?1 and f.following = 1")
public List<FollowData> getFollowing(int uid);
```

In Repository i have create declare the two method which is findByUserId is for get this list of follower of user.

And the second method getFollowing is for getting the list of following for that I define custom query using @Query annotation.

Service implementation layer of FollowData.

```
@Override
public List<FollowData> findByUserId(int uid) {
            List<FollowData> findByUserId = followRepo.findByUserId(uid);
            return findByUserId;
}

@Override
public List<FollowData> getFollowing(int uid) {
            List<FollowData> following = followRepo.getFollowing(uid);
            return following;
 }
```

Controller layer of FollowData.

```
@Autowired
private FollowDataService followDataService ;

      @GetMapping("{userId}/follower")
      public ResponseEntity<List<FollowData>> follower(@PathVariable("userId") int
userId){
            List<FollowData> list = followDataService.findByUserId(userId);
            return ResponseEntity.ok(list);

      }

      @GetMapping("{userId}/following")
      public ResponseEntity<?> following(@PathVariable("userId") int userId){
            List<FollowData> following = followDataService.getFollowing(userId);
            return ResponseEntity.ok(following);
      }
```

6.Show a list of users a person might know.

For this I have created one method in the FollowData repository layer.

```
@Query("from FollowData f where user.userId = ?1 or userId = ?1 and following = 1")
public List<FollowData> mightKnown(int uid);
```

Service implementation layer of FollowData.

```
@Override
public List<FollowData> mightKnown(int uid) {
            List<FollowData> mightKnown = followRepo.mightKnown(uid);
            return mightKnown;
}
```

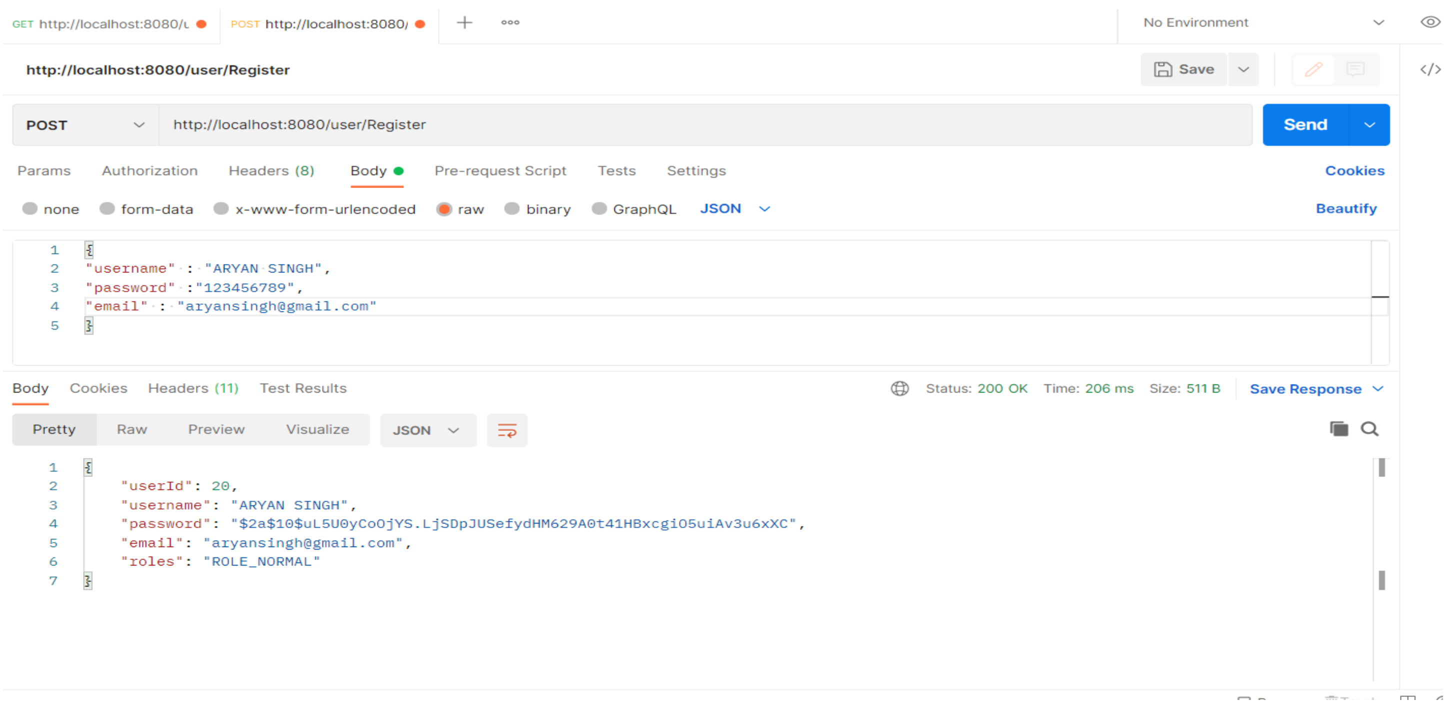
Controller Layer of FollowData.

```
@GetMapping("{userId}/mightKnown")
public ResponseEntity<?> mightKnown(@PathVariable("userId") int userId){

            //1st get a all following of user
            List<FollowData> following = followDataService.getFollowing(userId);
            List<FollowData> data = new ArrayList<FollowData>();

            //then the get the following list of user  following
            for (FollowData followData : following) {
                        //might known
                  List<FollowData> getknown =
            followDataService.mightKnown(followData.getUser().getUserId());
            //to get the list but not a user details bcoz its may following the user
back
                        for (FollowData followData2 : getknown ) {
                              if(followData2.getUserId() == userId) {

                              }else {
                                    data.add(followData2);
                              }
                        }
            }

            return ResponseEntity.ok(data);
}
```

Screenshots

Registration: -

Registration success:



http://localhost:8080/user/Register

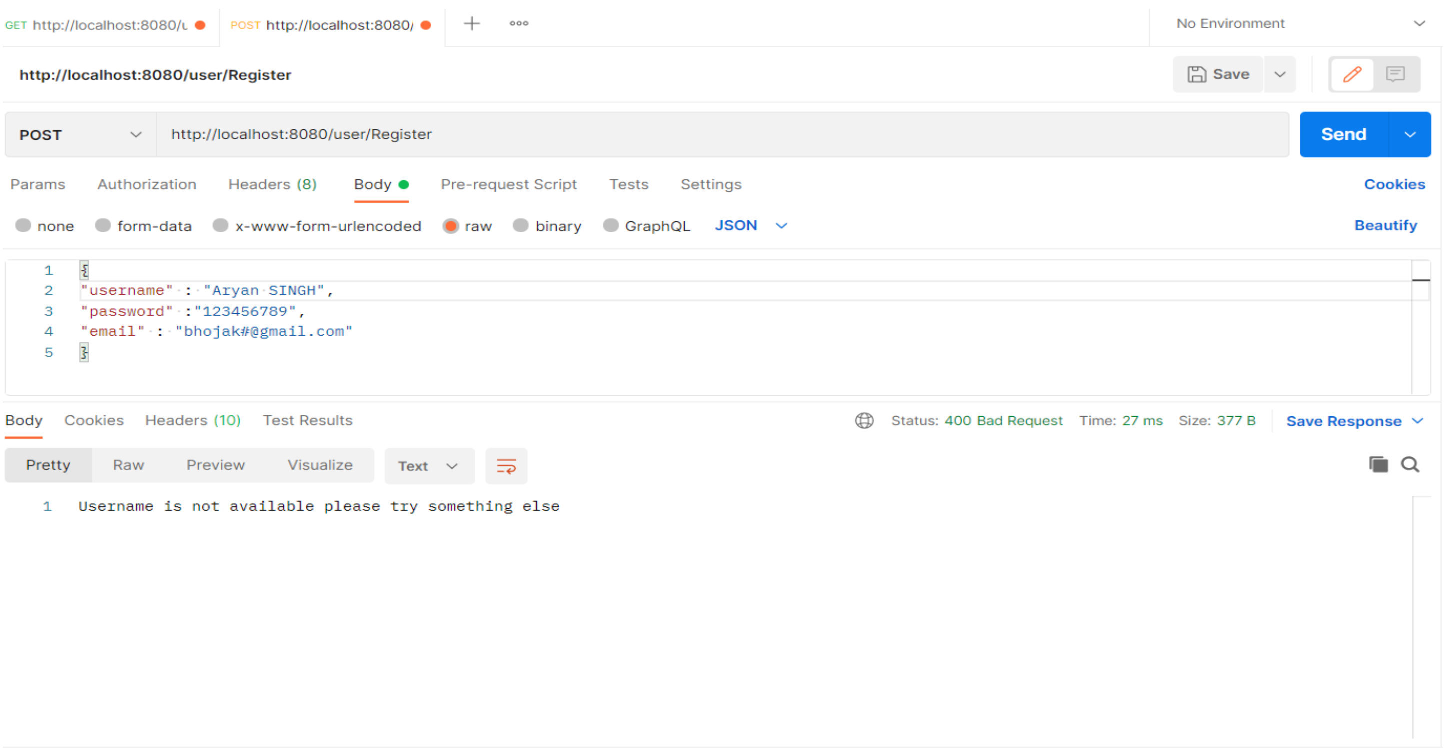
POST http://localhost:8080/user/Register

```
{
"username" : "ARYAN SINGH",
"password" :"123456789",
"email" : "aryansingh@gmail.com"
}
```

Status: 200 OK Time: 206 ms Size: 511 B

```
{
    "userId": 20,
    "username": "ARYAN SINGH",
    "password": "$2a$10$uL5U0yCoOjYS.LjSDpJUSefydHM629A0t41HBxcgiO5uiAv3u6xXC",
    "email": "aryansingh@gmail.com",
    "roles": "ROLE_NORMAL"
}
```

Registration fails:



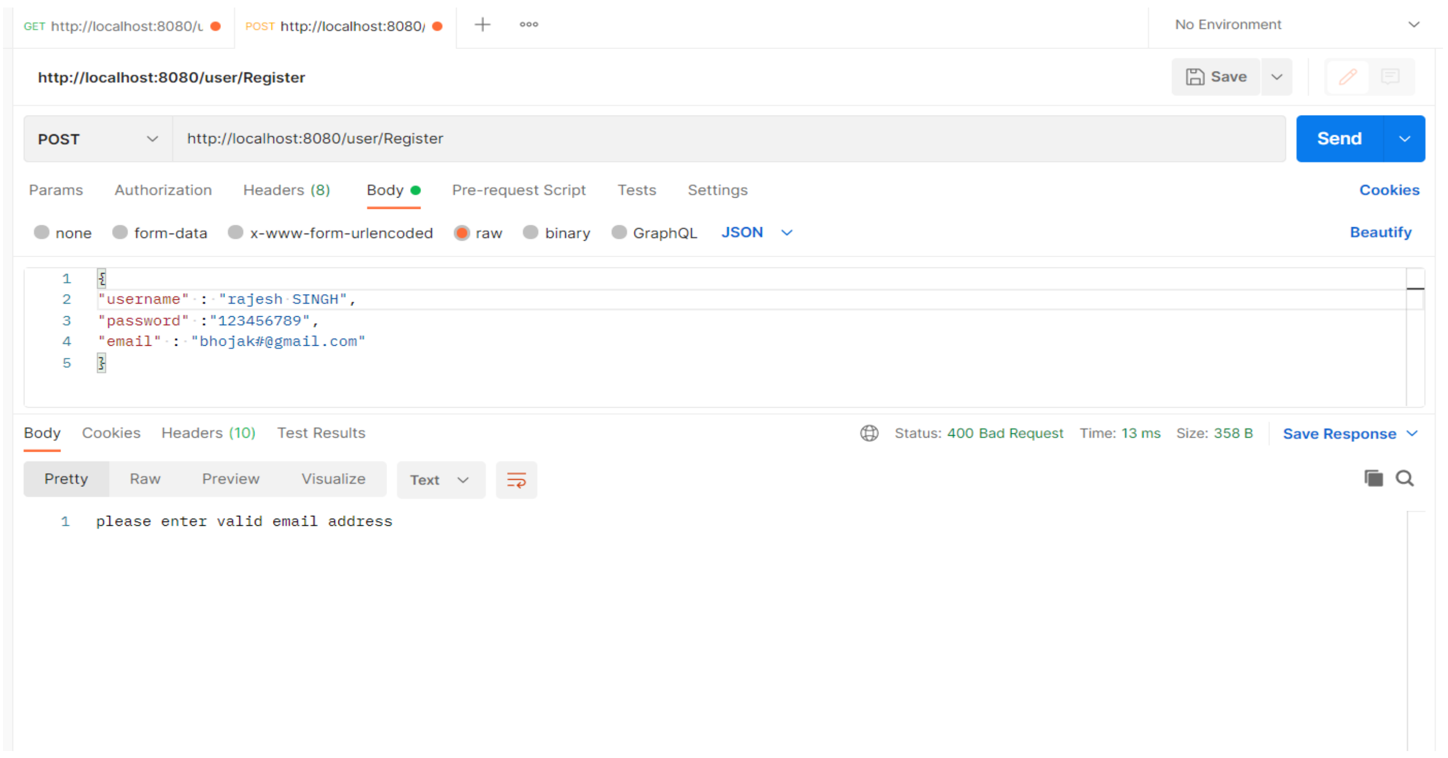
http://localhost:8080/user/Register

POST http://localhost:8080/user/Register

```
{
"username" : "Aryan SINGH",
"password" :"123456789",
"email" : "bhojak#@gmail.com"
}
```

Status: 400 Bad Request Time: 27 ms Size: 377 B

```
Username is not available please try something else
```



GET http://localhost:8080/u | POST http://localhost:8080/ | No Environment

http://localhost:8080/user/Register

POST http://localhost:8080/user/Register

Body: raw JSON

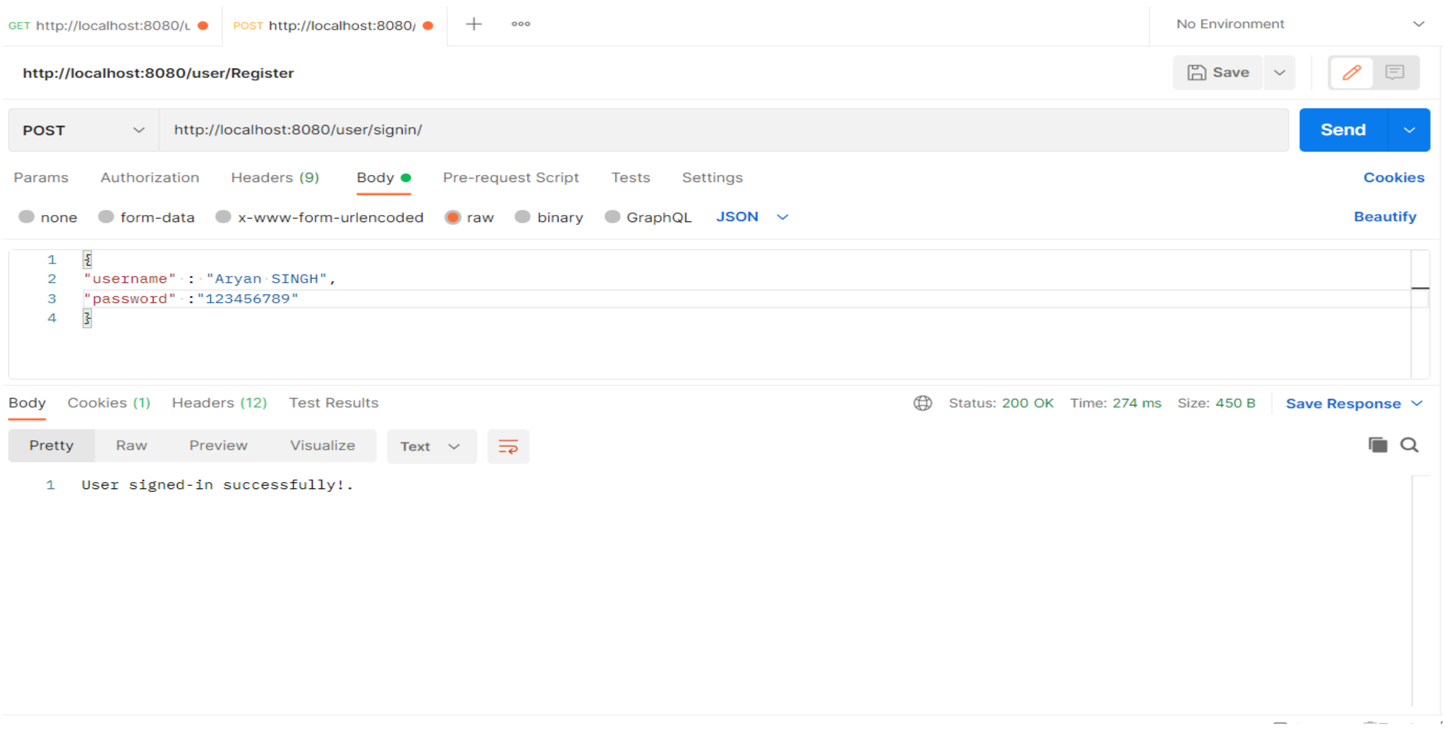
```
{
"username" : "rajesh SINGH",
"password" :"123456789",
"email" : "bhojak#@gmail.com"
}
```

Status: 400 Bad Request Time: 13 ms Size: 358 B

```
please enter valid email address
```

Login:

Login Success:



GET http://localhost:8080/u | POST http://localhost:8080/ | No Environment

http://localhost:8080/user/Register

POST http://localhost:8080/user/signin/

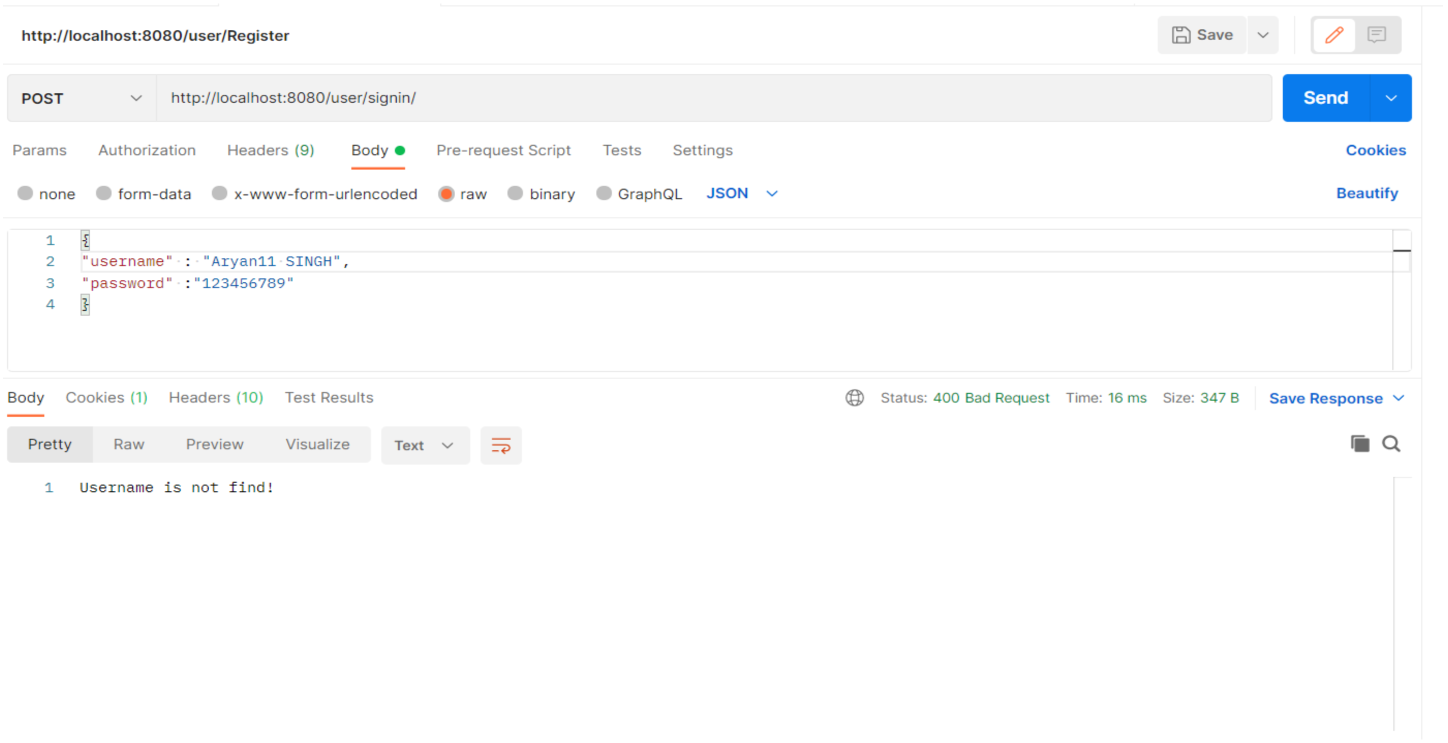
Body: raw JSON

```
{
"username" : "Aryan SINGH",
"password" :"123456789"
}
```

Status: 200 OK Time: 274 ms Size: 450 B

```
User signed-in successfully!.
```

Login fails:



http://localhost:8080/user/Register

POST http://localhost:8080/user/signin/

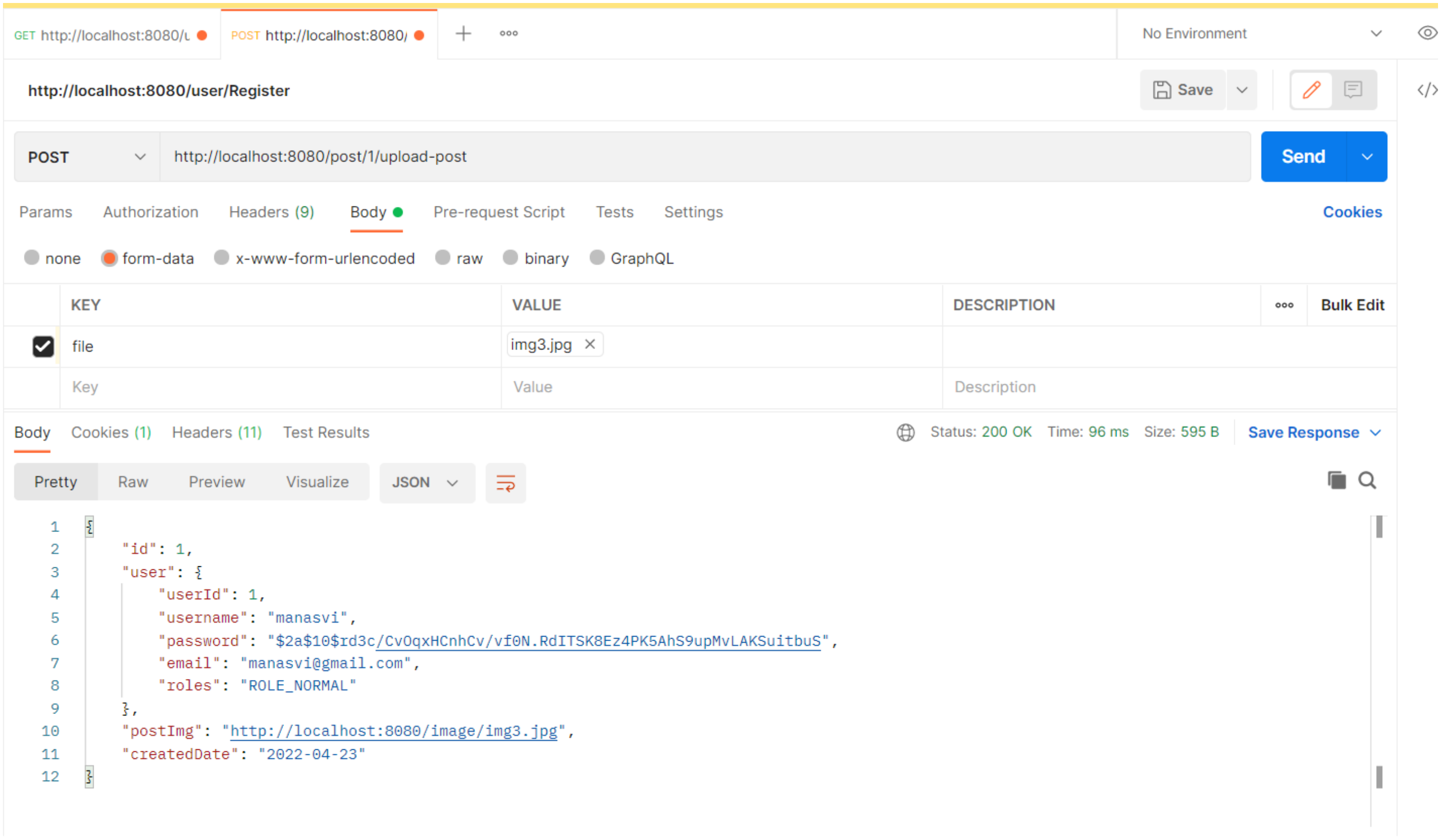
```
{
"username" : "Aryan11 SINGH",
"password" :"123456789"
}
```

Status: 400 Bad Request Time: 16 ms Size: 347 B

```
Username is not find!
```

Upload the post by user:

Upload success:



http://localhost:8080/user/Register

POST http://localhost:8080/post/1/upload-post

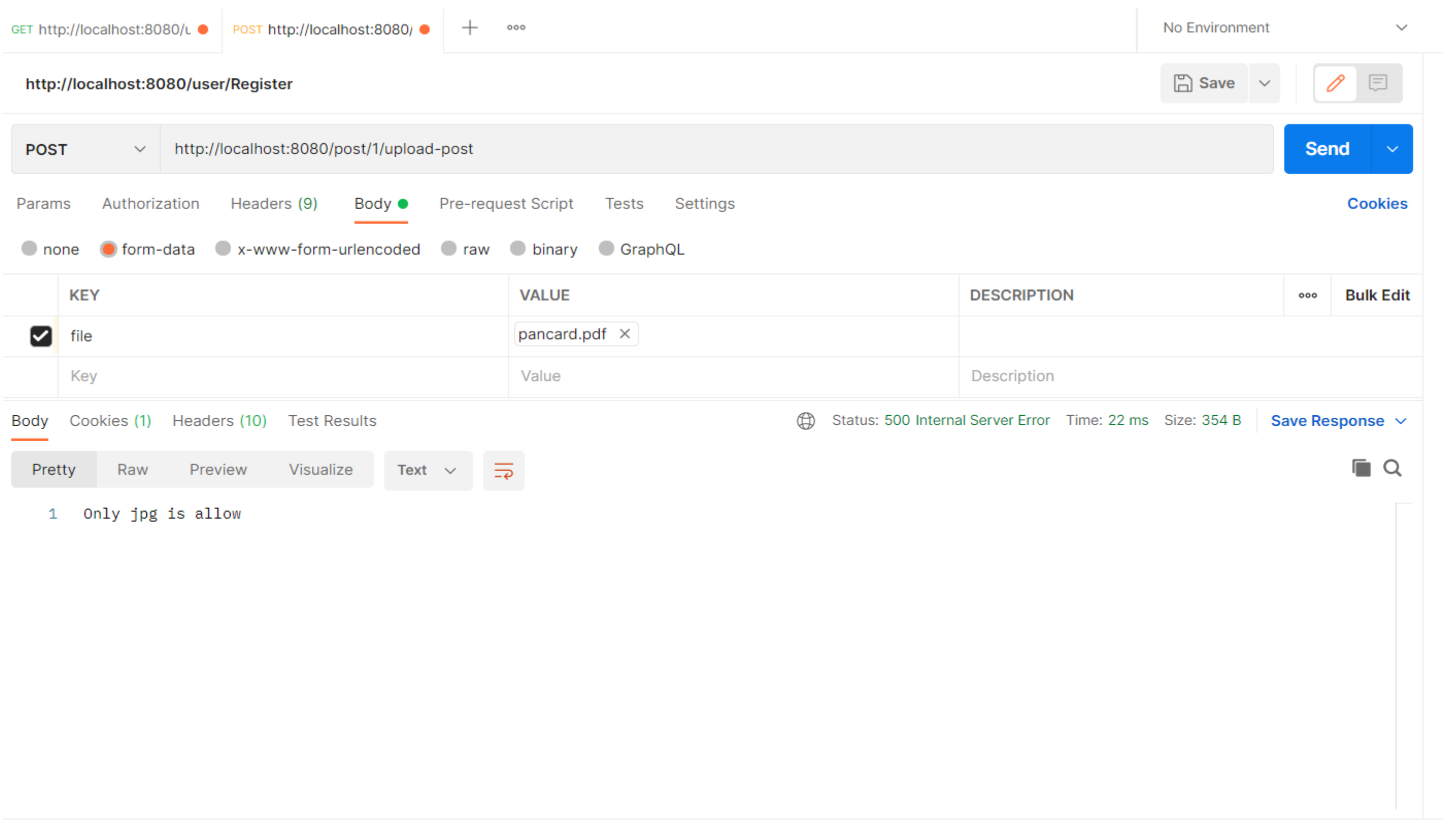
Body: form-data

KEY	VALUE	DESCRIPTION
file	img3.jpg	

Status: 200 OK Time: 96 ms Size: 595 B

```
{
    "id": 1,
    "user": {
        "userId": 1,
        "username": "manasvi",
        "password": "$2a$10$rd3c/CvOqxHCnhCv/vf0N.RdITSK8Ez4PK5AhS9upMvLAKSuitbuS",
        "email": "manasvi@gmail.com",
        "roles": "ROLE_NORMAL"
    },
    "postImg": "http://localhost:8080/image/img3.jpg",
    "createdDate": "2022-04-23"
}
```

Upload posts fails:



GET http://localhost:8080/u POST http://localhost:8080/

No Environment

http://localhost:8080/user/Register

POST http://localhost:8080/post/1/upload-post

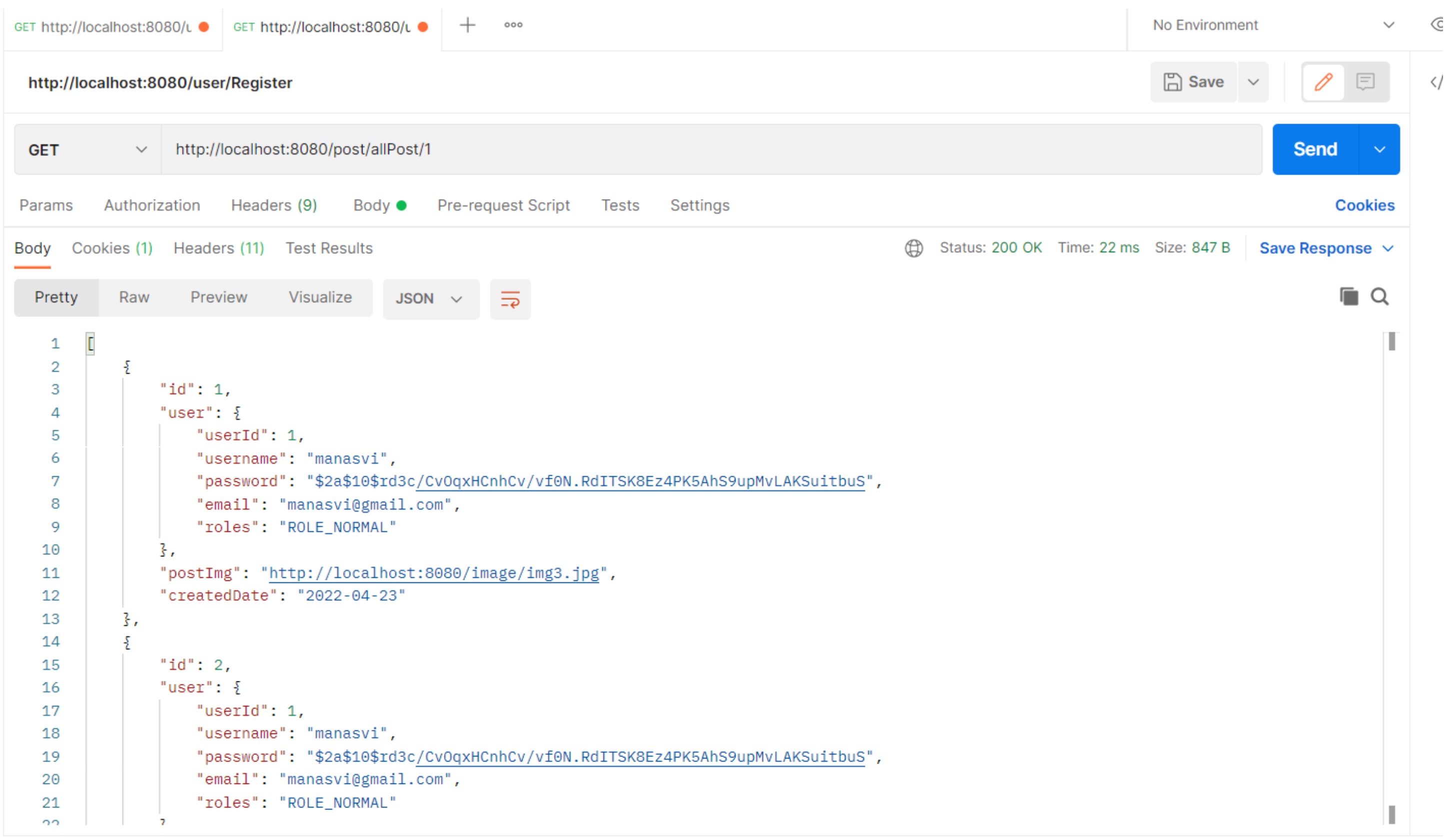
Body: form-data

KEY	VALUE	DESCRIPTION
file	pancard.pdf	

Status: 500 Internal Server Error Time: 22 ms Size: 354 B

```
Only jpg is allow
```

All posts of users:



GET http://localhost:8080/u GET http://localhost:8080/u

No Environment

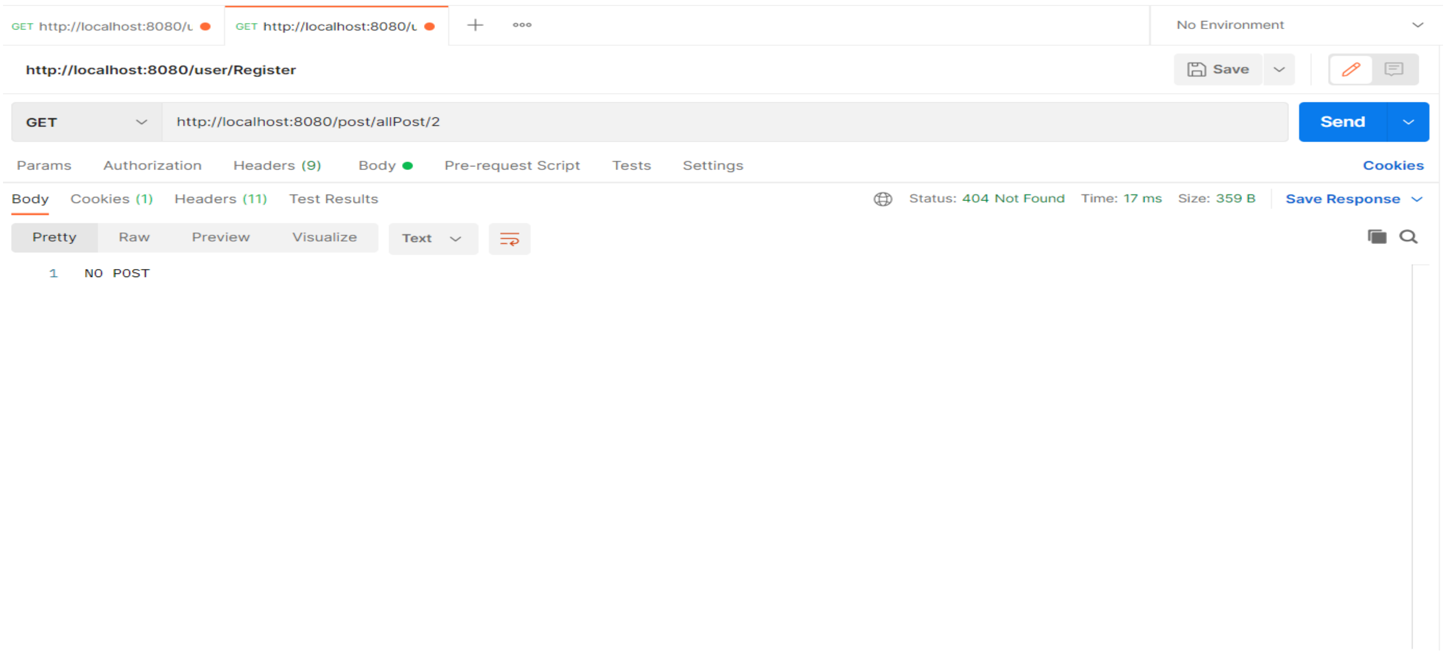
http://localhost:8080/user/Register

GET http://localhost:8080/post/allPost/1

Status: 200 OK Time: 22 ms Size: 847 B

```
[
    {
        "id": 1,
        "user": {
            "userId": 1,
            "username": "manasvi",
            "password": "$2a$10$rd3c/CvOqxHCnhCv/vf0N.RdITSK8Ez4PK5AhS9upMvLAKSuitbuS",
            "email": "manasvi@gmail.com",
            "roles": "ROLE_NORMAL"
        },
        "postImg": "http://localhost:8080/image/img3.jpg",
        "createdDate": "2022-04-23"
    },
    {
        "id": 2,
        "user": {
            "userId": 1,
            "username": "manasvi",
            "password": "$2a$10$rd3c/CvOqxHCnhCv/vf0N.RdITSK8Ez4PK5AhS9upMvLAKSuitbuS",
            "email": "manasvi@gmail.com",
            "roles": "ROLE_NORMAL"
```

When User have no posts:



GET http://localhost:8080/u ● GET http://localhost:8080/u ●

No Environment

http://localhost:8080/user/Register

GET http://localhost:8080/post/allPost/2

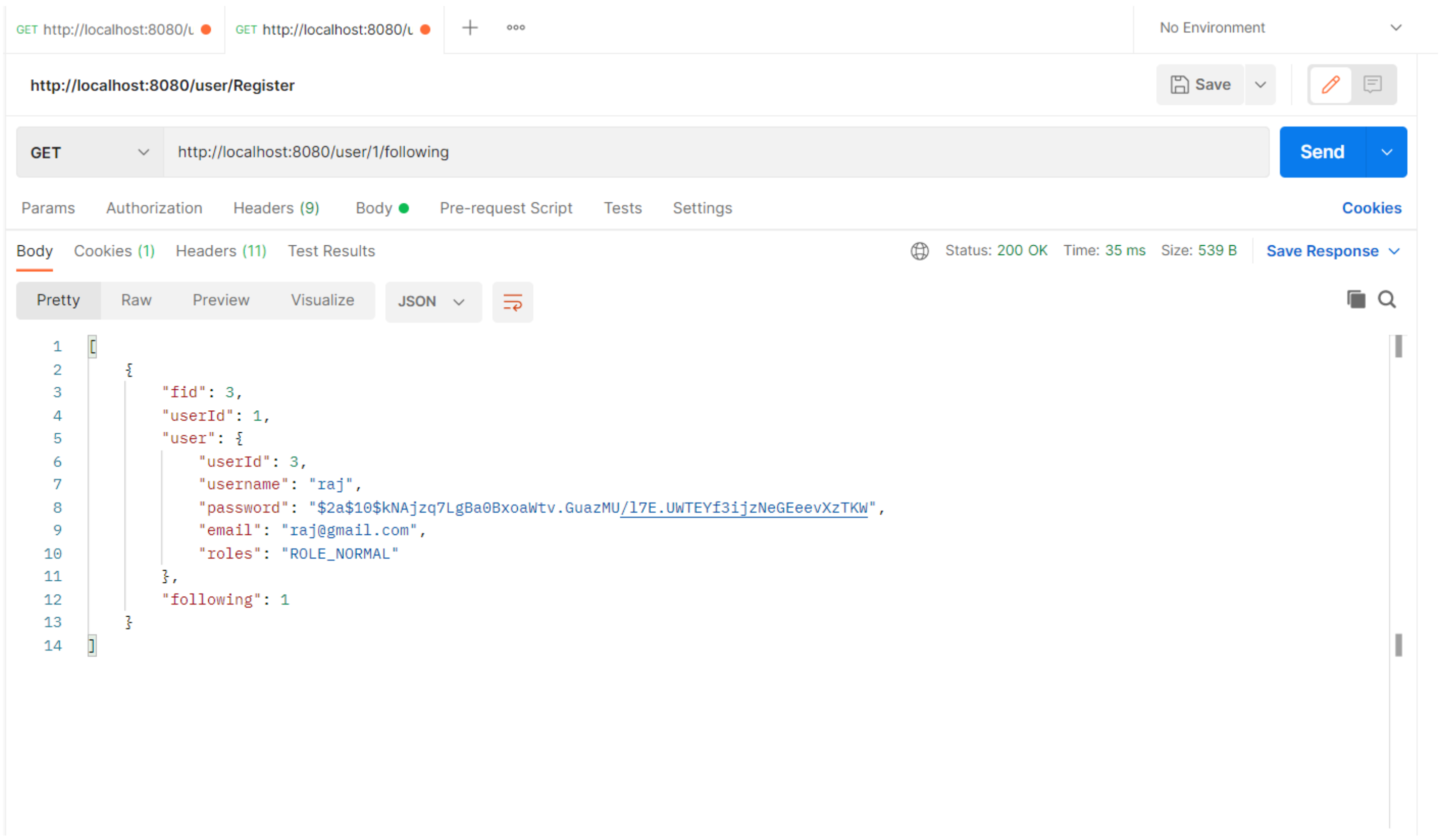
Params Authorization Headers (9) Body ● Pre-request Script Tests Settings Cookies

Body Cookies (1) Headers (11) Test Results Status: 404 Not Found Time: 17 ms Size: 359 B Save Response

Pretty Raw Preview Visualize Text

```
NO POST
```

Following List:



GET http://localhost:8080/u ● GET http://localhost:8080/u ●

No Environment

http://localhost:8080/user/Register

GET http://localhost:8080/user/1/following

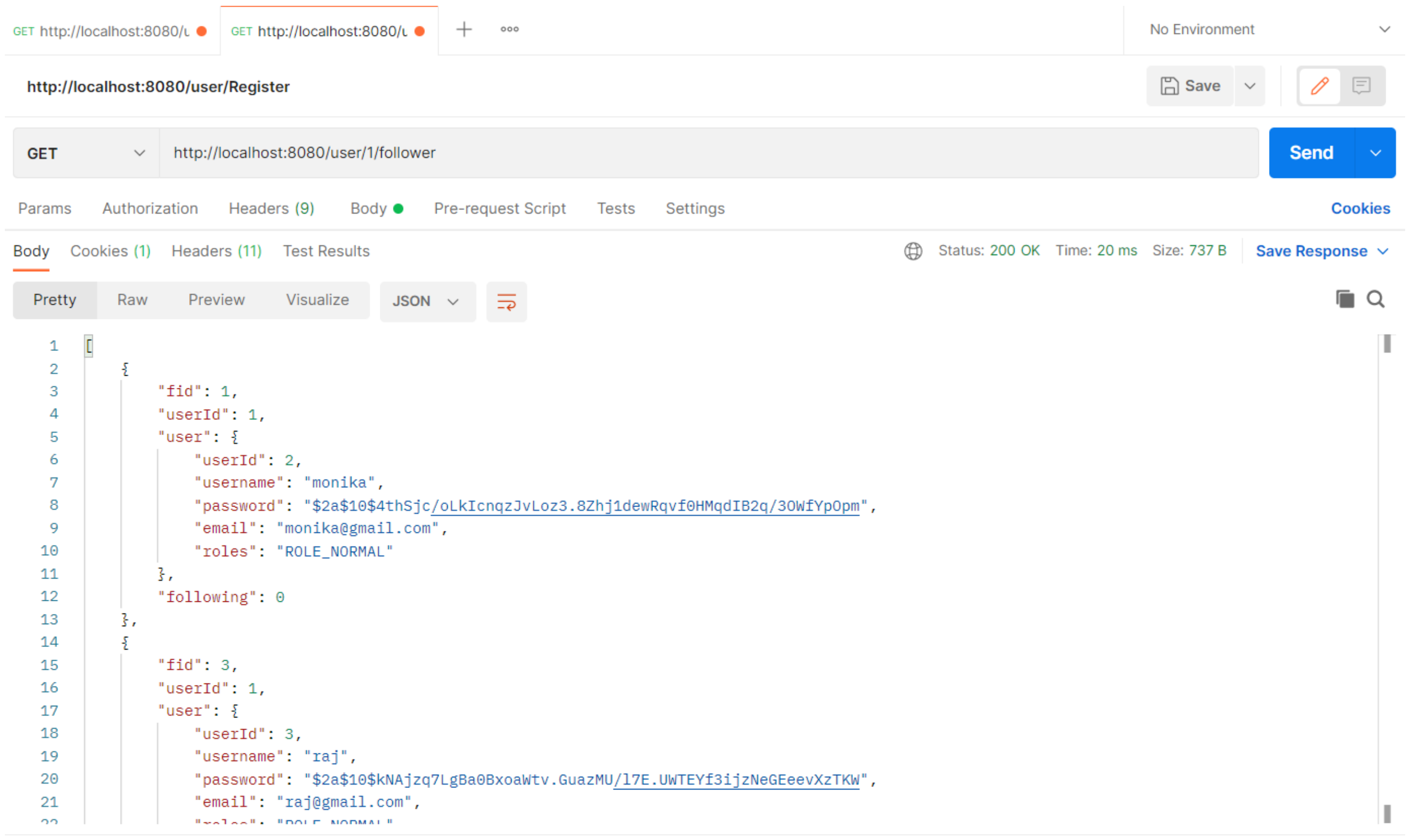
Params Authorization Headers (9) Body ● Pre-request Script Tests Settings Cookies

Body Cookies (1) Headers (11) Test Results Status: 200 OK Time: 35 ms Size: 539 B Save Response

Pretty Raw Preview Visualize JSON

```
[
    {
        "fid": 3,
        "userId": 1,
        "user": {
            "userId": 3,
            "username": "raj",
            "password": "$2a$10$kNAjzq7LgBa0BxoaWtv.GuazMU/l7E.UWTEYf3ijzNeGEeevXzTKW",
            "email": "raj@gmail.com",
            "roles": "ROLE_NORMAL"
        },
        "following": 1
    }
]
```

Followers List:



GET http://localhost:8080/u GET http://localhost:8080/u No Environment

http://localhost:8080/user/Register

GET http://localhost:8080/user/1/follower

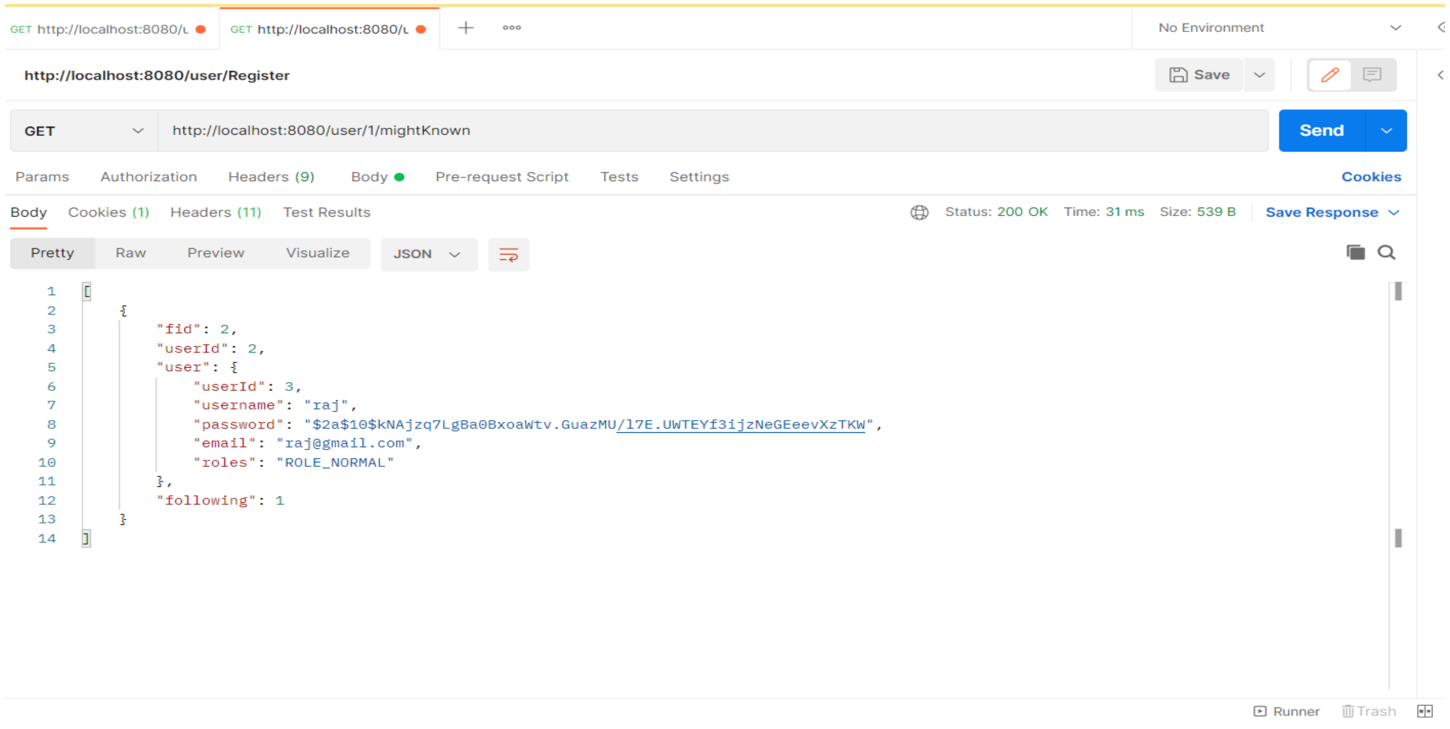
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body Cookies (1) Headers (11) Test Results Status: 200 OK Time: 20 ms Size: 737 B Save Response

Pretty Raw Preview Visualize JSON

```
[
    {
        "fid": 1,
        "userId": 1,
        "user": {
            "userId": 2,
            "username": "monika",
            "password": "$2a$10$4thSjc/oLkIcnqzJvLoz3.8Zhj1dewRqvf0HMqdIB2q/30WfYpOpm",
            "email": "monika@gmail.com",
            "roles": "ROLE_NORMAL"
        },
        "following": 0
    },
    {
        "fid": 3,
        "userId": 1,
        "user": {
            "userId": 3,
            "username": "raj",
            "password": "$2a$10$kNAjzq7LgBa0BxoaWtv.GuazMU/l7E.UWTEYf3ijzNeGEeevXzTKW",
            "email": "raj@gmail.com",
```

User might know people list:



GET http://localhost:8080/u GET http://localhost:8080/u No Environment

http://localhost:8080/user/Register

GET http://localhost:8080/user/1/mightKnown

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body Cookies (1) Headers (11) Test Results Status: 200 OK Time: 31 ms Size: 539 B Save Response

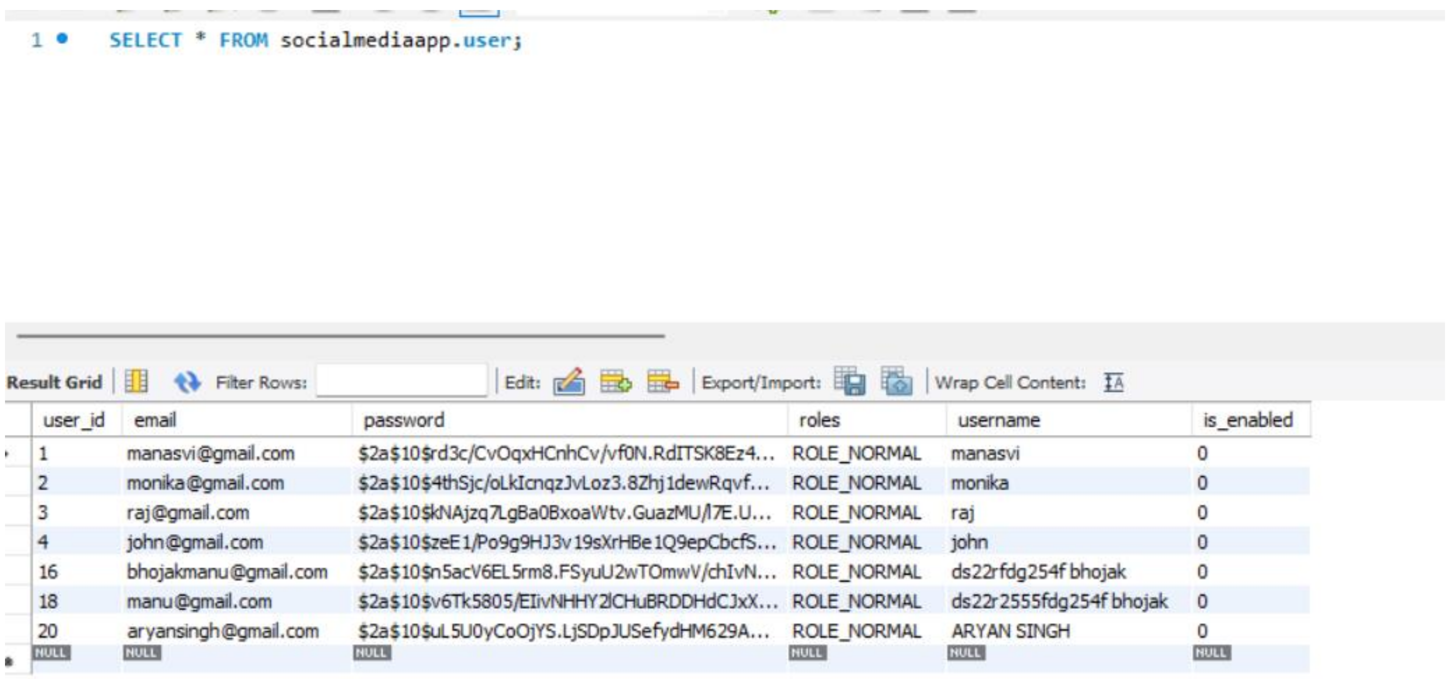
Pretty Raw Preview Visualize JSON

```
[
    {
        "fid": 2,
        "userId": 2,
        "user": {
            "userId": 3,
            "username": "raj",
            "password": "$2a$10$kNAjzq7LgBa0BxoaWtv.GuazMU/l7E.UWTEYf3ijzNeGEeevXzTKW",
            "email": "raj@gmail.com",
            "roles": "ROLE_NORMAL"
        },
        "following": 1
    }
]
```

Runner Trash

DataBase

User DataBase:

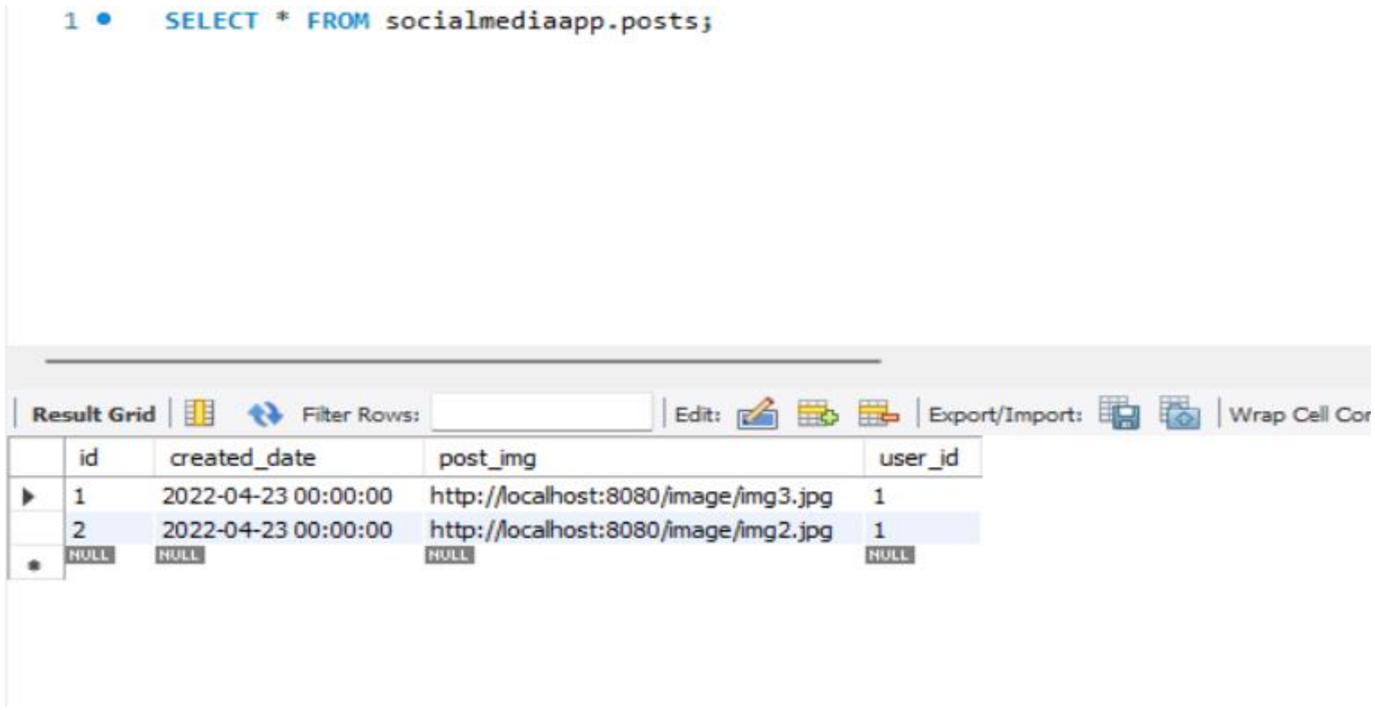


```
SELECT * FROM socialmediaapp.user;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

user_id	email	password	roles	username	is_enabled
1	manasvi@gmail.com	$2a$10$rd3c/CvOqxHCnhCv/vf0N.RdITSK8Ez4...	ROLE_NORMAL	manasvi	0
2	monika@gmail.com	$2a$10$4thSjc/oLkIcnqzJvLoz3.8Zhj1dewRqvf...	ROLE_NORMAL	monika	0
3	raj@gmail.com	$2a$10$kNAjzq7LgBa0BxoaWtv.GuazMU/l7E.U...	ROLE_NORMAL	raj	0
4	john@gmail.com	$2a$10$zeE1/Po9g9HJ3v19sXrHBe1Q9epCbcfS...	ROLE_NORMAL	john	0
16	bhojakmanu@gmail.com	$2a$10$n5acV6EL5rm8.FSyuU2wTOmwV/chIvN...	ROLE_NORMAL	ds22rfdg254f bhojak	0
18	manu@gmail.com	$2a$10$v6Tk5805/EIivNHHY2lCHuBRDDHdCJxX...	ROLE_NORMAL	ds22r2555fdg254f bhojak	0
20	aryansingh@gmail.com	$2a$10$uL5U0yCoOjYS.LjSDpJUSefydHM629A...	ROLE_NORMAL	ARYAN SINGH	0
NULL	NULL	NULL	NULL	NULL	NULL

Post DataBase:

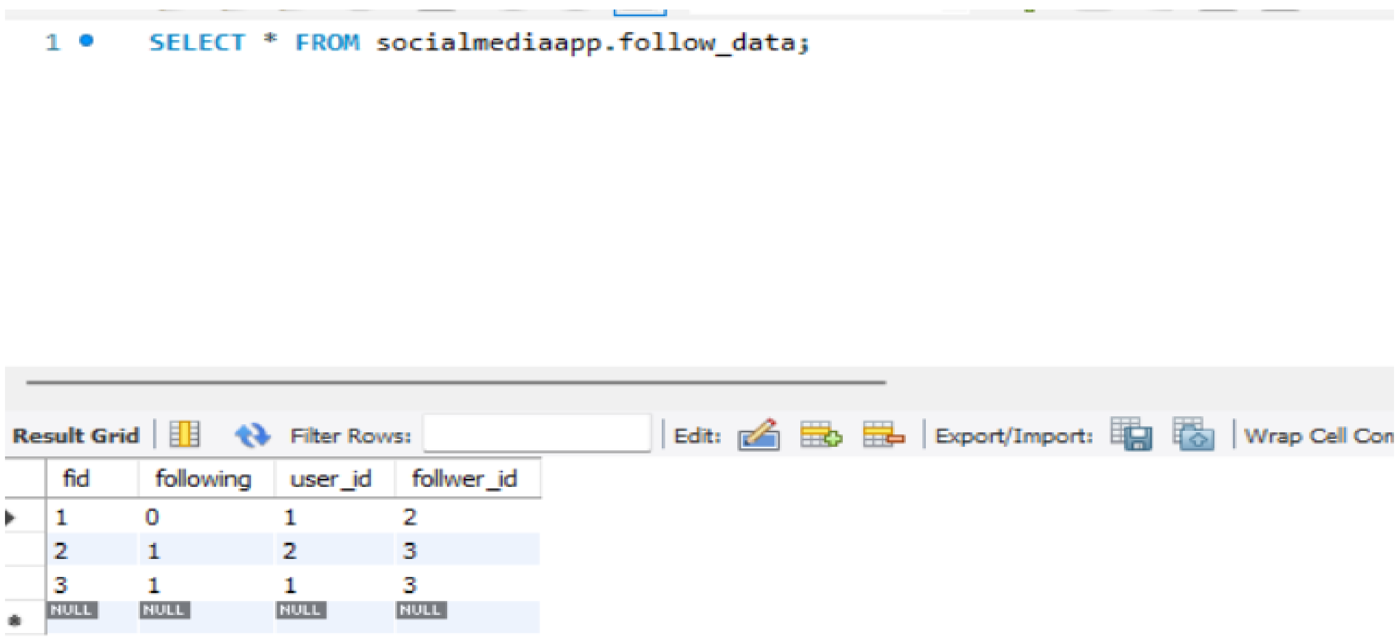


```
SELECT * FROM socialmediaapp.posts;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cor

id	created_date	post_img	user_id
1	2022-04-23 00:00:00	http://localhost:8080/image/img3.jpg	1
2	2022-04-23 00:00:00	http://localhost:8080/image/img2.jpg	1
NULL	NULL	NULL	NULL

FollowData DataBase:



```
SELECT * FROM socialmediaapp.follow_data;
```

fid	following	user_id	follwer_id
1	0	1	2
2	1	2	3
3	1	1	3
NULL	NULL	NULL	NULL

Thank You