Web API Design with SpringBoot Week 1 Coding Assignment

Points possible: 70

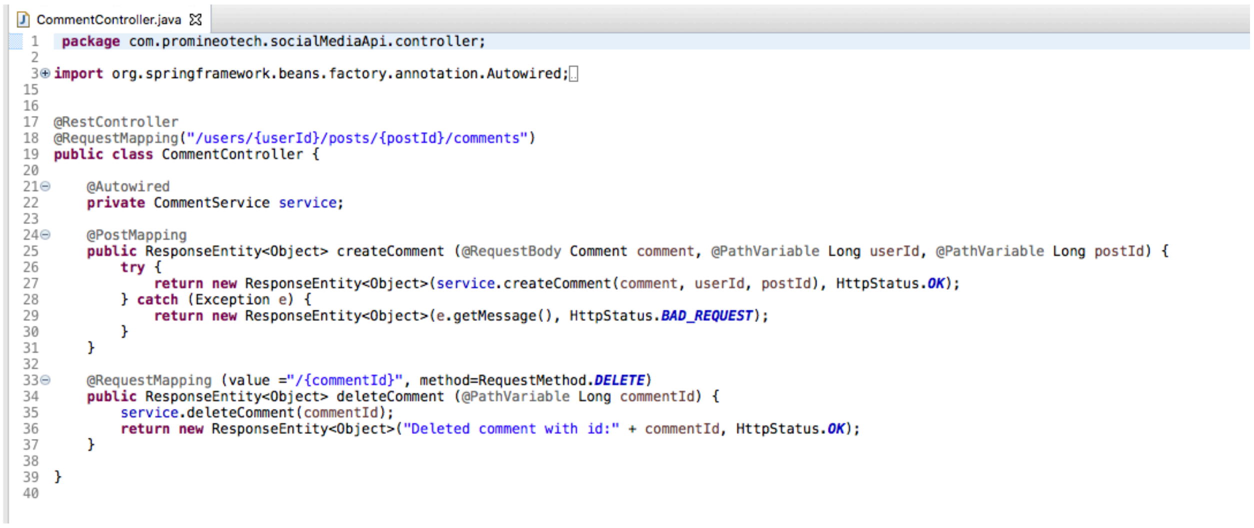
Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Lastly, in the Learning Management System, click the "Add Submission" button and paste the URL to your GitHub repository.

Coding Steps:

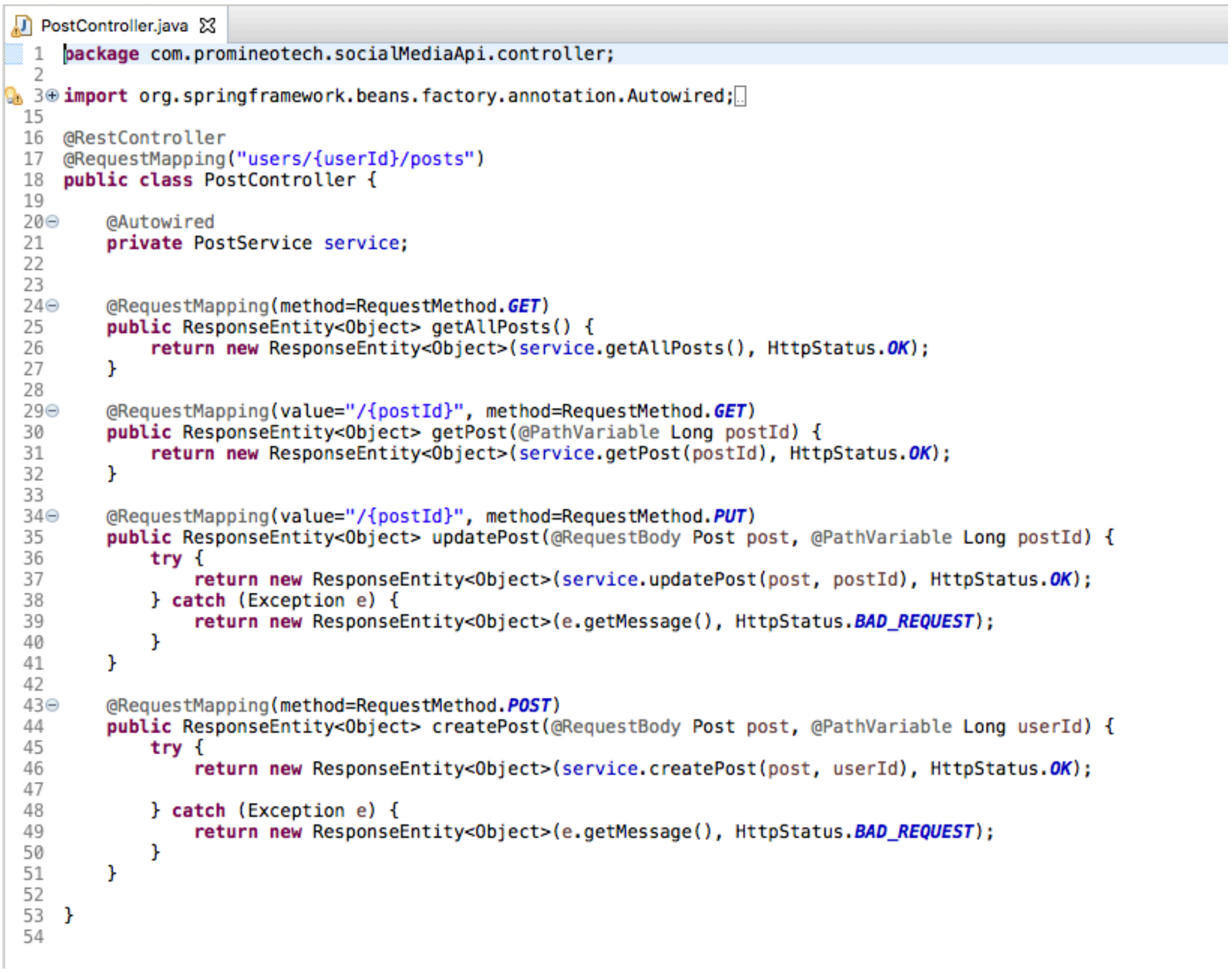
Follow the tutorial in the homework section to build an API. Paste screenshots of your code and your postman requests and responses to show the API works. Push your project to GitHub and paste the link below.

Screenshots of Code:



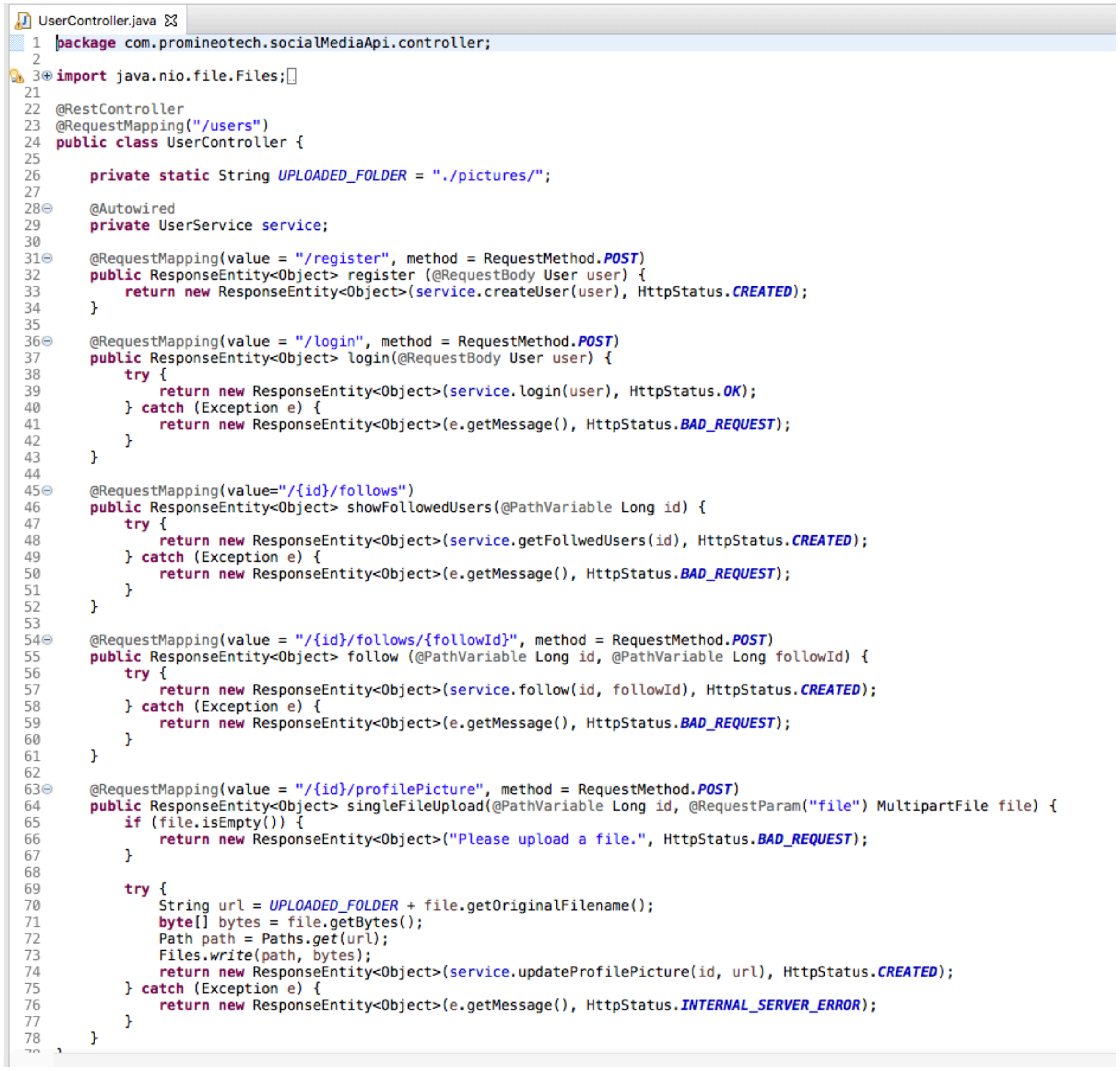
CommentController.java

```java
package com.promineotech.socialMediaApi.controller;

import org.springframework.beans.factory.annotation.Autowired;


@RestController
@RequestMapping("/users/{userId}/posts/{postId}/comments")
public class CommentController {

    @Autowired
    private CommentService service;

    @PostMapping
    public ResponseEntity<Object> createComment (@RequestBody Comment comment, @PathVariable Long userId, @PathVariable Long postId) {
        try {
            return new ResponseEntity<Object>(service.createComment(comment, userId, postId), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping (value ="/{commentId}", method=RequestMethod.DELETE)
    public ResponseEntity<Object> deleteComment (@PathVariable Long commentId) {
        service.deleteComment(commentId);
        return new ResponseEntity<Object>("Deleted comment with id:" + commentId, HttpStatus.OK);
    }

}

```

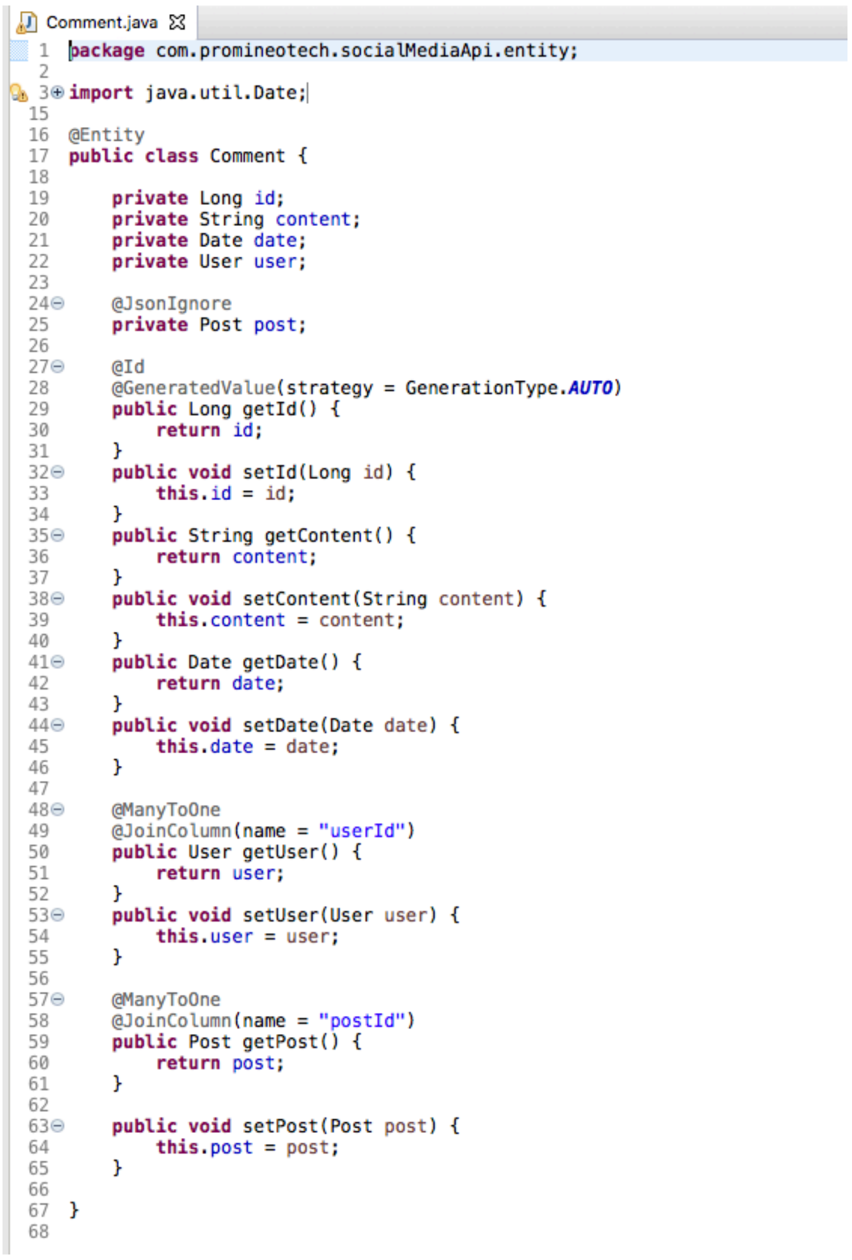


PostController.java

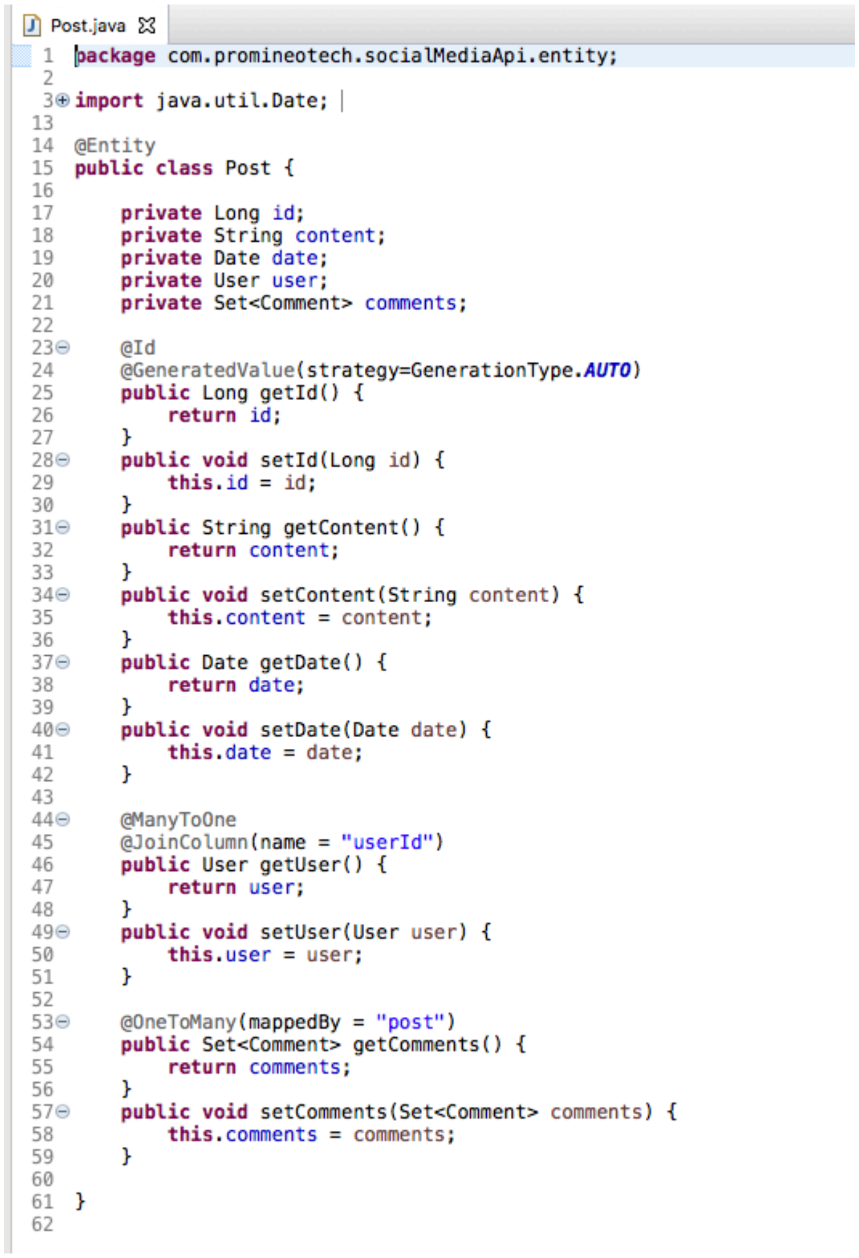
```java
package com.promineotech.socialMediaApi.controller;

import org.springframework.beans.factory.annotation.Autowired;

@RestController
@RequestMapping("users/{userId}/posts")
public class PostController {

    @Autowired
    private PostService service;


    @RequestMapping(method=RequestMethod.GET)
    public ResponseEntity<Object> getAllPosts() {
        return new ResponseEntity<Object>(service.getAllPosts(), HttpStatus.OK);
    }

    @RequestMapping(value="/{postId}", method=RequestMethod.GET)
    public ResponseEntity<Object> getPost(@PathVariable Long postId) {
        return new ResponseEntity<Object>(service.getPost(postId), HttpStatus.OK);
    }

    @RequestMapping(value="/{postId}", method=RequestMethod.PUT)
    public ResponseEntity<Object> updatePost(@RequestBody Post post, @PathVariable Long postId) {
        try {
            return new ResponseEntity<Object>(service.updatePost(post, postId), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(method=RequestMethod.POST)
    public ResponseEntity<Object> createPost(@RequestBody Post post, @PathVariable Long userId) {
        try {
            return new ResponseEntity<Object>(service.createPost(post, userId), HttpStatus.OK);

        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
        }
    }

}

```


UserController.java

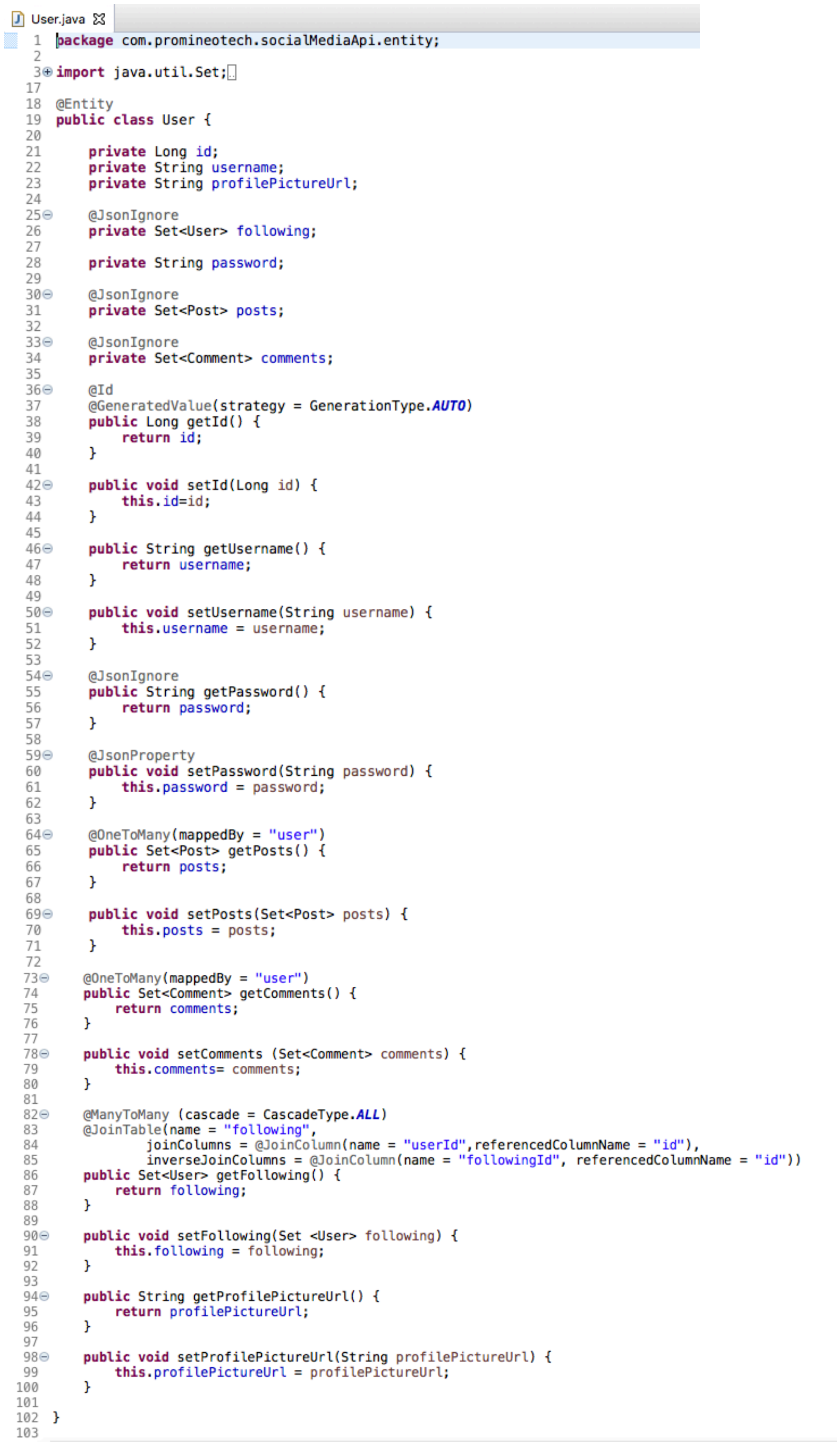
```java
package com.promineotech.socialMediaApi.controller;

import java.nio.file.Files;

@RestController
@RequestMapping("/users")
public class UserController {

    private static String UPLOADED_FOLDER = "./pictures/";

    @Autowired
    private UserService service;

    @RequestMapping(value = "/register", method = RequestMethod.POST)
    public ResponseEntity<Object> register (@RequestBody User user) {
        return new ResponseEntity<Object>(service.createUser(user), HttpStatus.CREATED);
    }

    @RequestMapping(value = "/login", method = RequestMethod.POST)
    public ResponseEntity<Object> login(@RequestBody User user) {
        try {
            return new ResponseEntity<Object>(service.login(user), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(value="/{id}/follows")
    public ResponseEntity<Object> showFollowedUsers(@PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.getFollwedUsers(id), HttpStatus.CREATED);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(value = "/{id}/follows/{followId}", method = RequestMethod.POST)
    public ResponseEntity<Object> follow (@PathVariable Long id, @PathVariable Long followId) {
        try {
            return new ResponseEntity<Object>(service.follow(id, followId), HttpStatus.CREATED);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(value = "/{id}/profilePicture", method = RequestMethod.POST)
    public ResponseEntity<Object> singleFileUpload(@PathVariable Long id, @RequestParam("file") MultipartFile file) {
        if (file.isEmpty()) {
            return new ResponseEntity<Object>("Please upload a file.", HttpStatus.BAD_REQUEST);
        }

        try {
            String url = UPLOADED_FOLDER + file.getOriginalFilename();
            byte[] bytes = file.getBytes();
            Path path = Paths.get(url);
            Files.write(path, bytes);
            return new ResponseEntity<Object>(service.updateProfilePicture(id, url), HttpStatus.CREATED);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
```


Comment.java

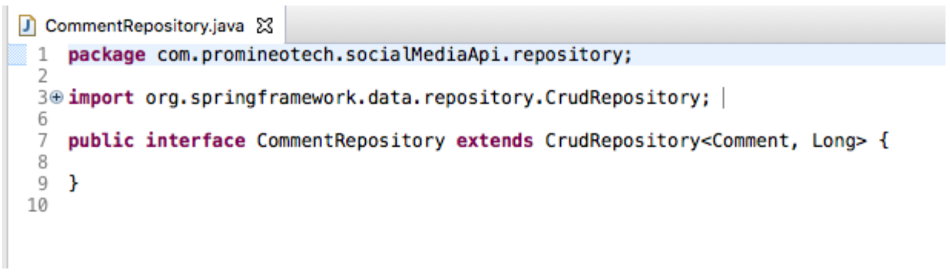
```java
package com.promineotech.socialMediaApi.entity;

import java.util.Date;

@Entity
public class Comment {

    private Long id;
    private String content;
    private Date date;
    private User user;

    @JsonIgnore
    private Post post;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }

    @ManyToOne
    @JoinColumn(name = "userId")
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
    }

    @ManyToOne
    @JoinColumn(name = "postId")
    public Post getPost() {
        return post;
    }

    public void setPost(Post post) {
        this.post = post;
    }

}
```


Post.java

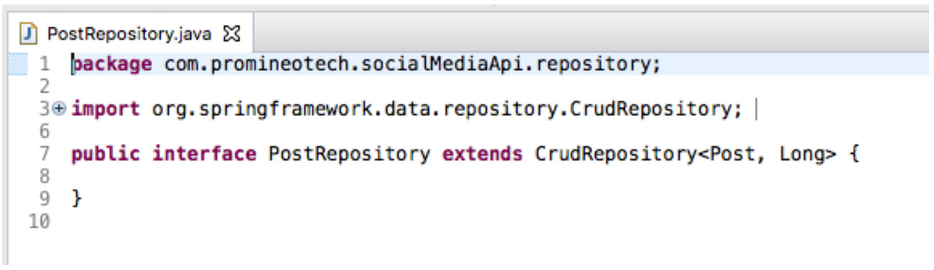
```java
package com.promineotech.socialMediaApi.entity;

import java.util.Date;

@Entity
public class Post {

    private Long id;
    private String content;
    private Date date;
    private User user;
    private Set<Comment> comments;

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }

    @ManyToOne
    @JoinColumn(name = "userId")
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
    }

    @OneToMany(mappedBy = "post")
    public Set<Comment> getComments() {
        return comments;
    }
    public void setComments(Set<Comment> comments) {
        this.comments = comments;
    }

}
```



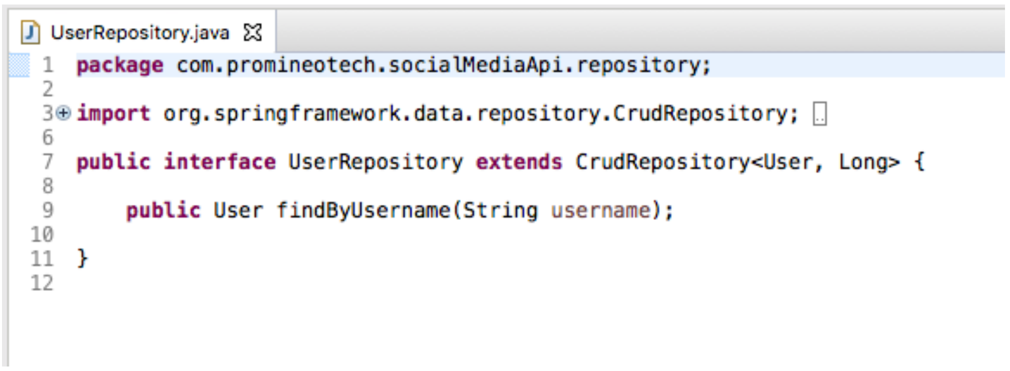
```java
package com.promineotech.socialMediaApi.entity;

import java.util.Set;

@Entity
public class User {

    private Long id;
    private String username;
    private String profilePictureUrl;

    @JsonIgnore
    private Set<User> following;

    private String password;

    @JsonIgnore
    private Set<Post> posts;

    @JsonIgnore
    private Set<Comment> comments;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id=id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @JsonIgnore
    public String getPassword() {
        return password;
    }

    @JsonProperty
    public void setPassword(String password) {
        this.password = password;
    }

    @OneToMany(mappedBy = "user")
    public Set<Post> getPosts() {
        return posts;
    }

    public void setPosts(Set<Post> posts) {
        this.posts = posts;
    }

    @OneToMany(mappedBy = "user")
    public Set<Comment> getComments() {
        return comments;
    }

    public void setComments (Set<Comment> comments) {
        this.comments= comments;
    }

    @ManyToMany (cascade = CascadeType.ALL)
    @JoinTable(name = "following",
            joinColumns = @JoinColumn(name = "userId",referencedColumnName = "id"),
            inverseJoinColumns = @JoinColumn(name = "followingId", referencedColumnName = "id"))
    public Set<User> getFollowing() {
        return following;
    }

    public void setFollowing(Set <User> following) {
        this.following = following;
    }

    public String getProfilePictureUrl() {
        return profilePictureUrl;
    }

    public void setProfilePictureUrl(String profilePictureUrl) {
        this.profilePictureUrl = profilePictureUrl;
    }

}
```


CommentRepository.java

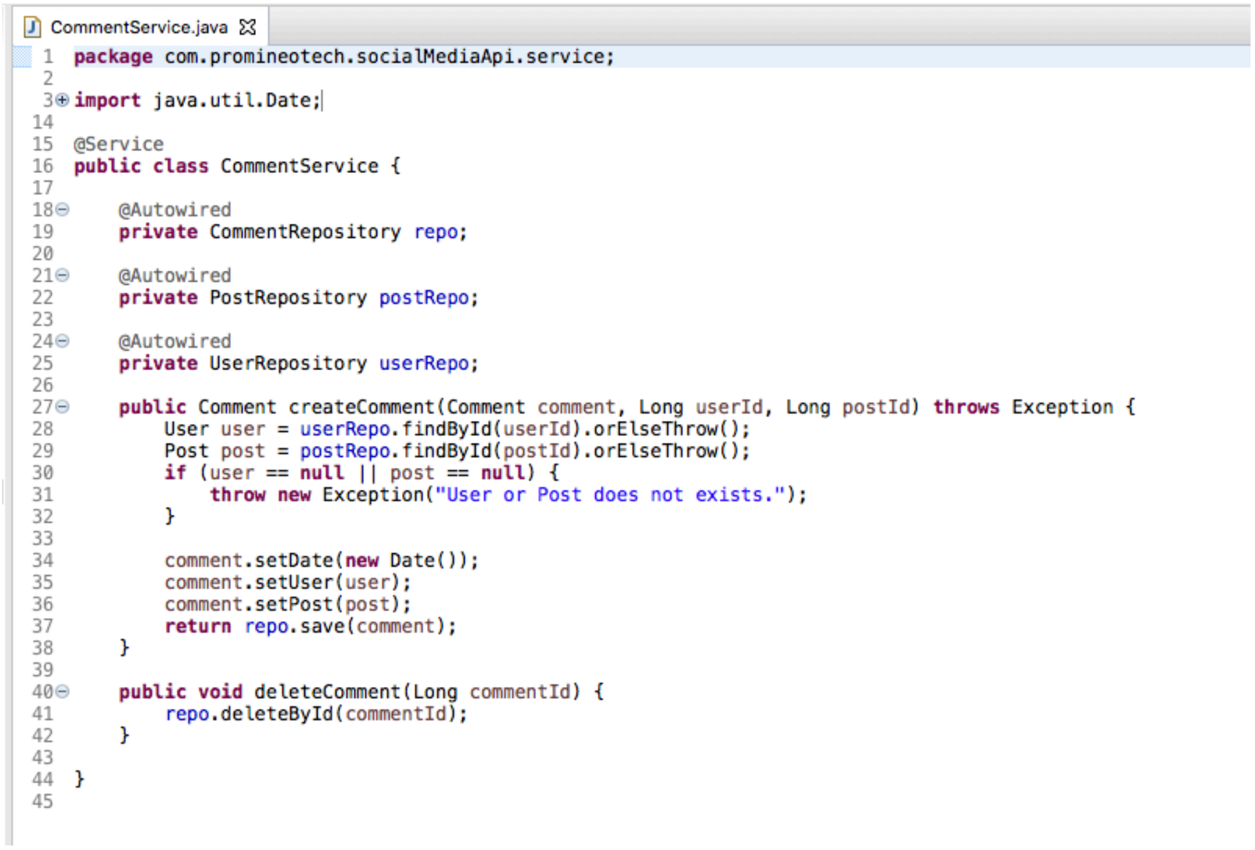
```java
package com.promineotech.socialMediaApi.repository;

import org.springframework.data.repository.CrudRepository;

public interface CommentRepository extends CrudRepository<Comment, Long> {

}

```


PostRepository.java

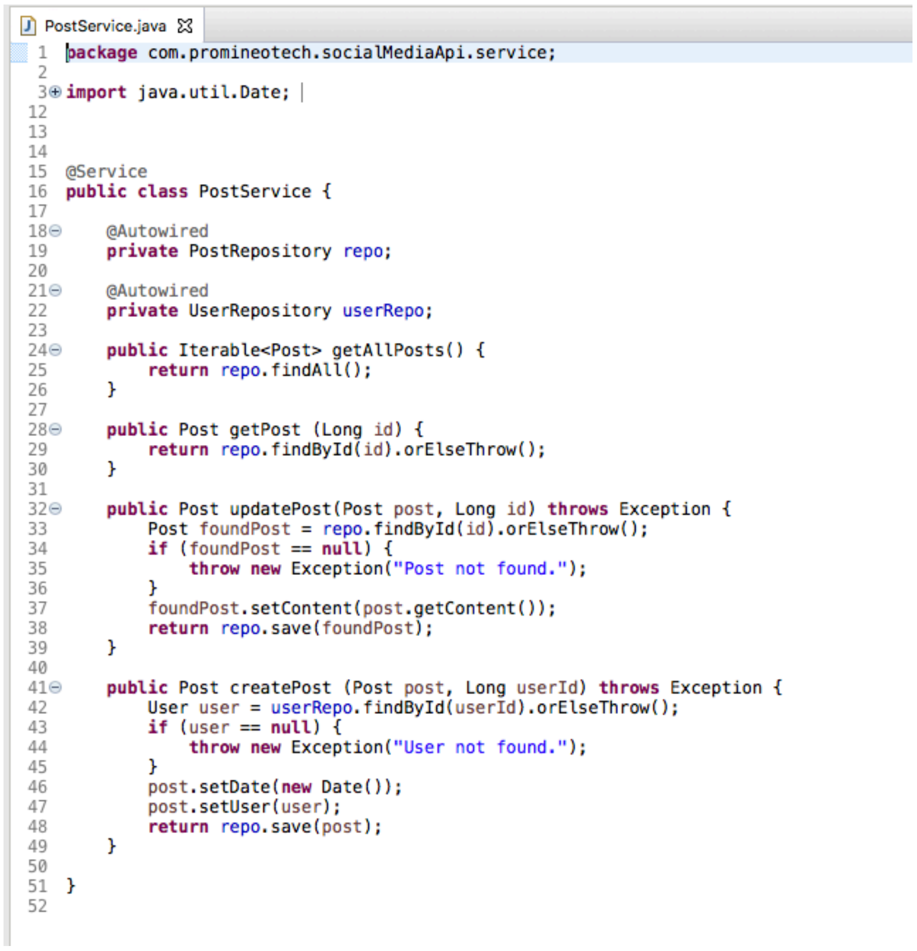
```java
package com.promineotech.socialMediaApi.repository;

import org.springframework.data.repository.CrudRepository;

public interface PostRepository extends CrudRepository<Post, Long> {

}

```


UserRepository.java

```java
package com.promineotech.socialMediaApi.repository;

import org.springframework.data.repository.CrudRepository;

public interface UserRepository extends CrudRepository<User, Long> {

    public User findByUsername(String username);

}

```

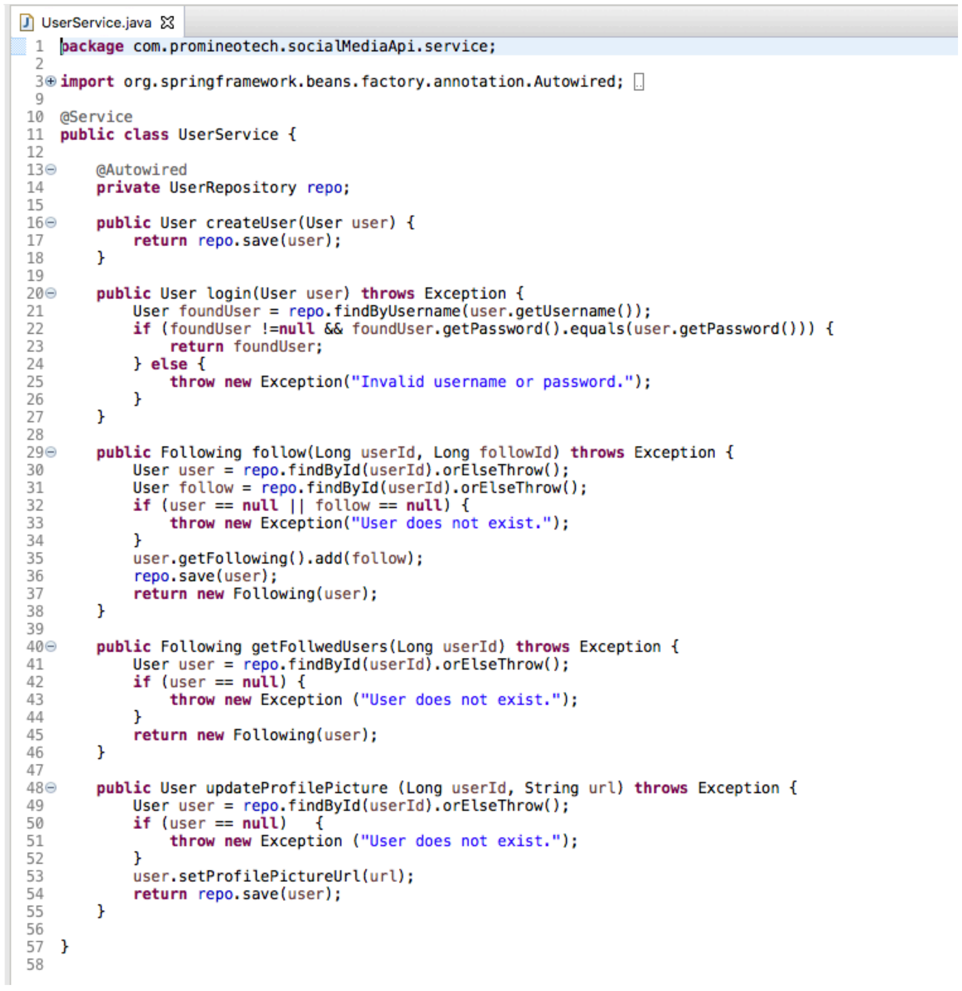

CommentService.java

```java
package com.promineotech.socialMediaApi.service;

import java.util.Date;

@Service
public class CommentService {

    @Autowired
    private CommentRepository repo;

    @Autowired
    private PostRepository postRepo;

    @Autowired
    private UserRepository userRepo;

    public Comment createComment(Comment comment, Long userId, Long postId) throws Exception {
        User user = userRepo.findById(userId).orElseThrow();
        Post post = postRepo.findById(postId).orElseThrow();
        if (user == null || post == null) {
            throw new Exception("User or Post does not exists.");
        }

        comment.setDate(new Date());
        comment.setUser(user);
        comment.setPost(post);
        return repo.save(comment);
    }

    public void deleteComment(Long commentId) {
        repo.deleteById(commentId);
    }

}

```



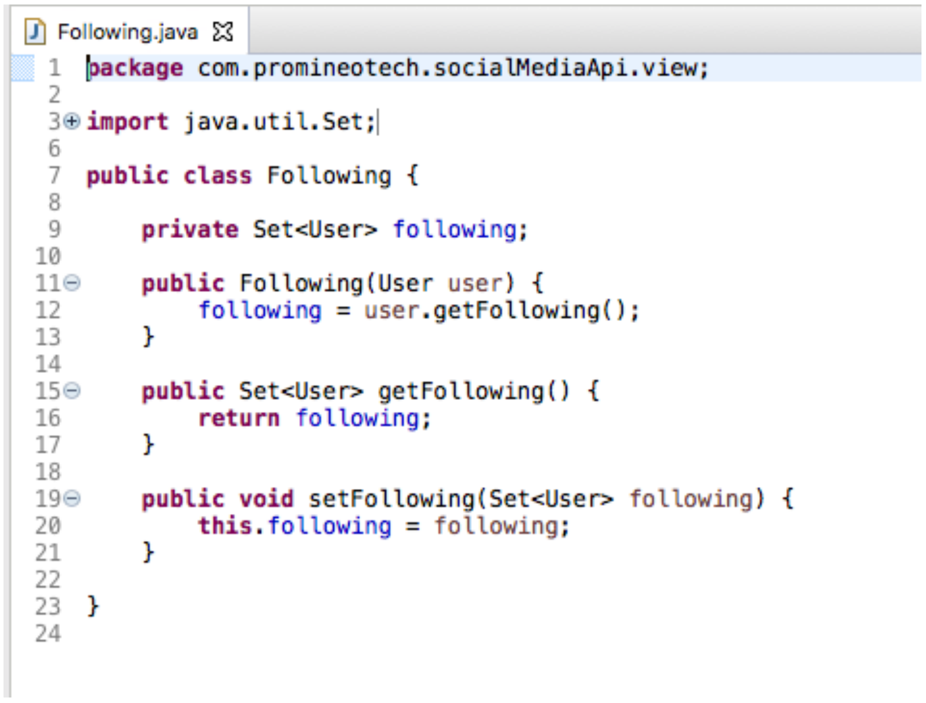
PostService.java

```java
package com.promineotech.socialMediaApi.service;

import java.util.Date;



@Service
public class PostService {

    @Autowired
    private PostRepository repo;

    @Autowired
    private UserRepository userRepo;

    public Iterable<Post> getAllPosts() {
        return repo.findAll();
    }

    public Post getPost (Long id) {
        return repo.findById(id).orElseThrow();
    }

    public Post updatePost(Post post, Long id) throws Exception {
        Post foundPost = repo.findById(id).orElseThrow();
        if (foundPost == null) {
            throw new Exception("Post not found.");
        }
        foundPost.setContent(post.getContent());
        return repo.save(foundPost);
    }

    public Post createPost (Post post, Long userId) throws Exception {
        User user = userRepo.findById(userId).orElseThrow();
        if (user == null) {
            throw new Exception("User not found.");
        }
        post.setDate(new Date());
        post.setUser(user);
        return repo.save(post);
    }

}
```

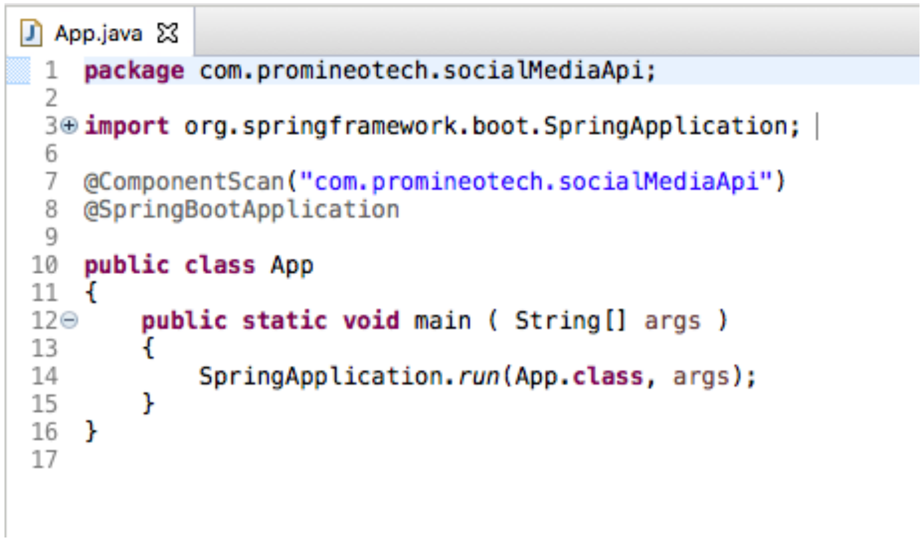


UserService.java

```java
package com.promineotech.socialMediaApi.service;

import org.springframework.beans.factory.annotation.Autowired;

@Service
public class UserService {

    @Autowired
    private UserRepository repo;

    public User createUser(User user) {
        return repo.save(user);
    }

    public User login(User user) throws Exception {
        User foundUser = repo.findByUsername(user.getUsername());
        if (foundUser !=null && foundUser.getPassword().equals(user.getPassword())) {
            return foundUser;
        } else {
            throw new Exception("Invalid username or password.");
        }
    }

    public Following follow(Long userId, Long followId) throws Exception {
        User user = repo.findById(userId).orElseThrow();
        User follow = repo.findById(userId).orElseThrow();
        if (user == null || follow == null) {
            throw new Exception("User does not exist.");
        }
        user.getFollowing().add(follow);
        repo.save(user);
        return new Following(user);
    }

    public Following getFollwedUsers(Long userId) throws Exception {
        User user = repo.findById(userId).orElseThrow();
        if (user == null) {
            throw new Exception ("User does not exist.");
        }
        return new Following(user);
    }

    public User updateProfilePicture (Long userId, String url) throws Exception {
        User user = repo.findById(userId).orElseThrow();
        if (user == null)   {
            throw new Exception ("User does not exist.");
        }
        user.setProfilePictureUrl(url);
        return repo.save(user);
    }

}
```

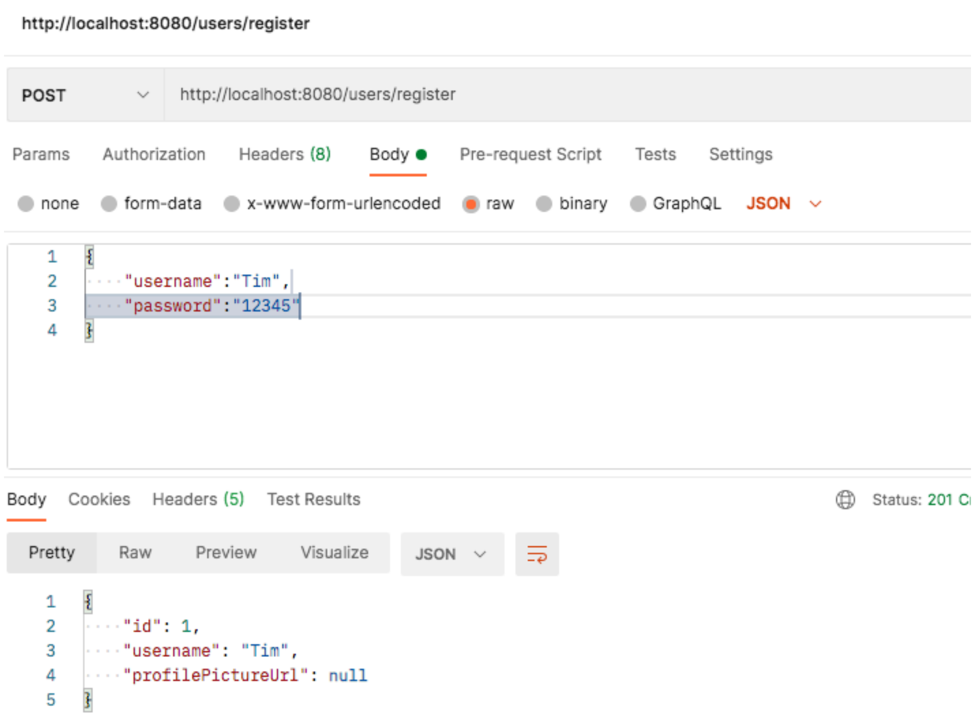

Following.java

```java
package com.promineotech.socialMediaApi.view;

import java.util.Set;

public class Following {

    private Set<User> following;

    public Following(User user) {
        following = user.getFollowing();
    }

    public Set<User> getFollowing() {
        return following;
    }

    public void setFollowing(Set<User> following) {
        this.following = following;
    }

}
```


App.java

```java
package com.promineotech.socialMediaApi;

import org.springframework.boot.SpringApplication;

@ComponentScan("com.promineotech.socialMediaApi")
@SpringBootApplication

public class App
{
    public static void main ( String[] args )
    {
        SpringApplication.run(App.class, args);
    }
}
```

Postman Screenshots:

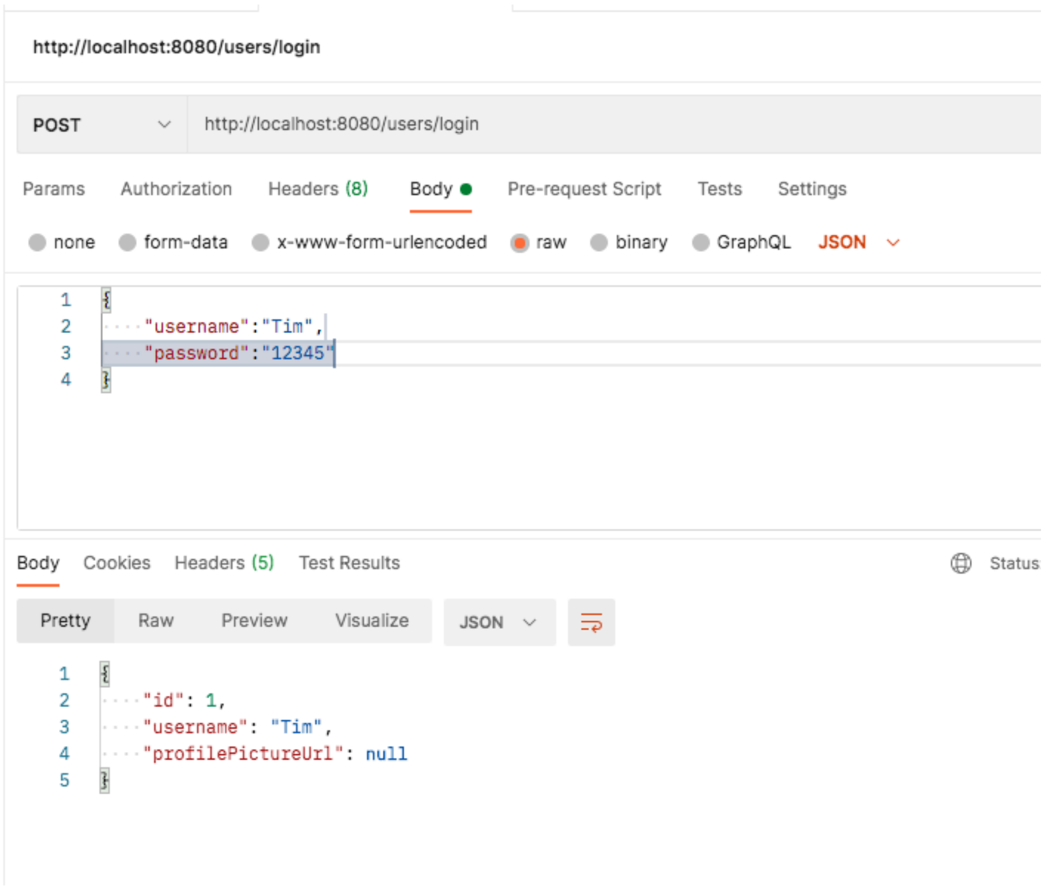


http://localhost:8080/users/register

POST http://localhost:8080/users/register

```
{
    "username":"Tim",
    "password":"12345"
}
```

```
{
    "id": 1,
    "username": "Tim",
    "profilePictureUrl": null
}
```

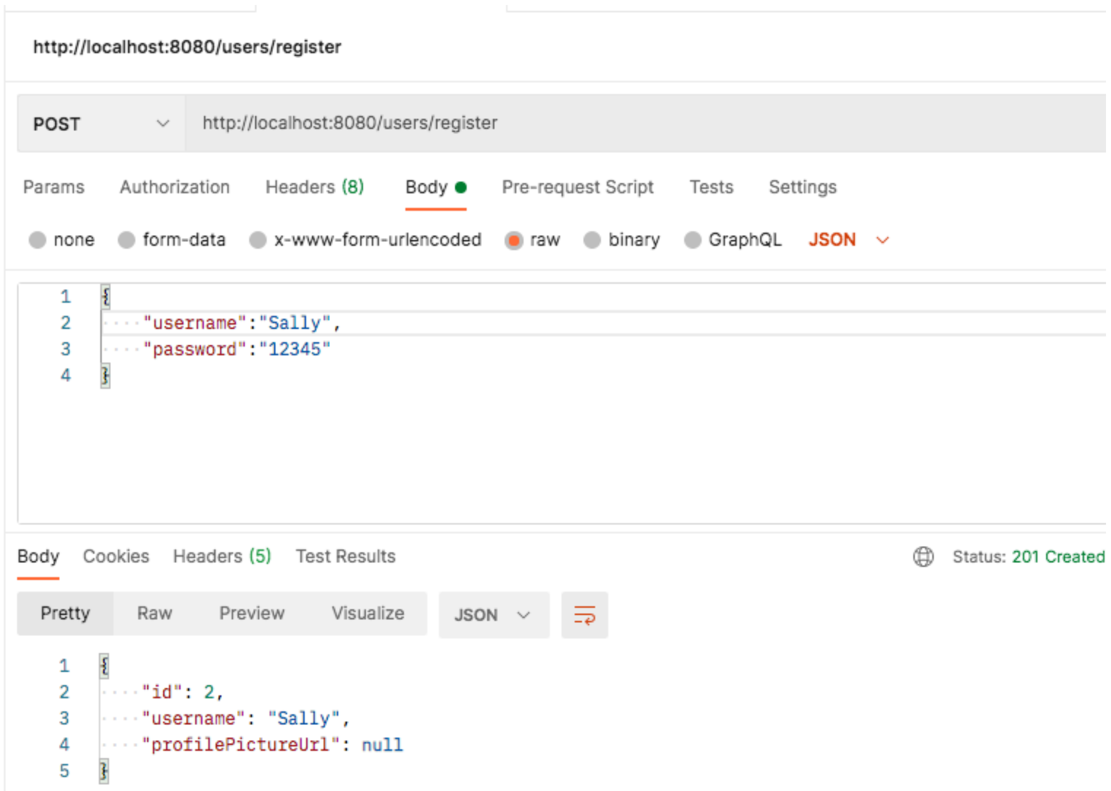


http://localhost:8080/users/login

POST http://localhost:8080/users/login

```
{
    "username":"Tim",
    "password":"12345"
}
```

```
{
    "id": 1,
    "username": "Tim",
    "profilePictureUrl": null
}
```



http://localhost:8080/users/register

POST http://localhost:8080/users/register

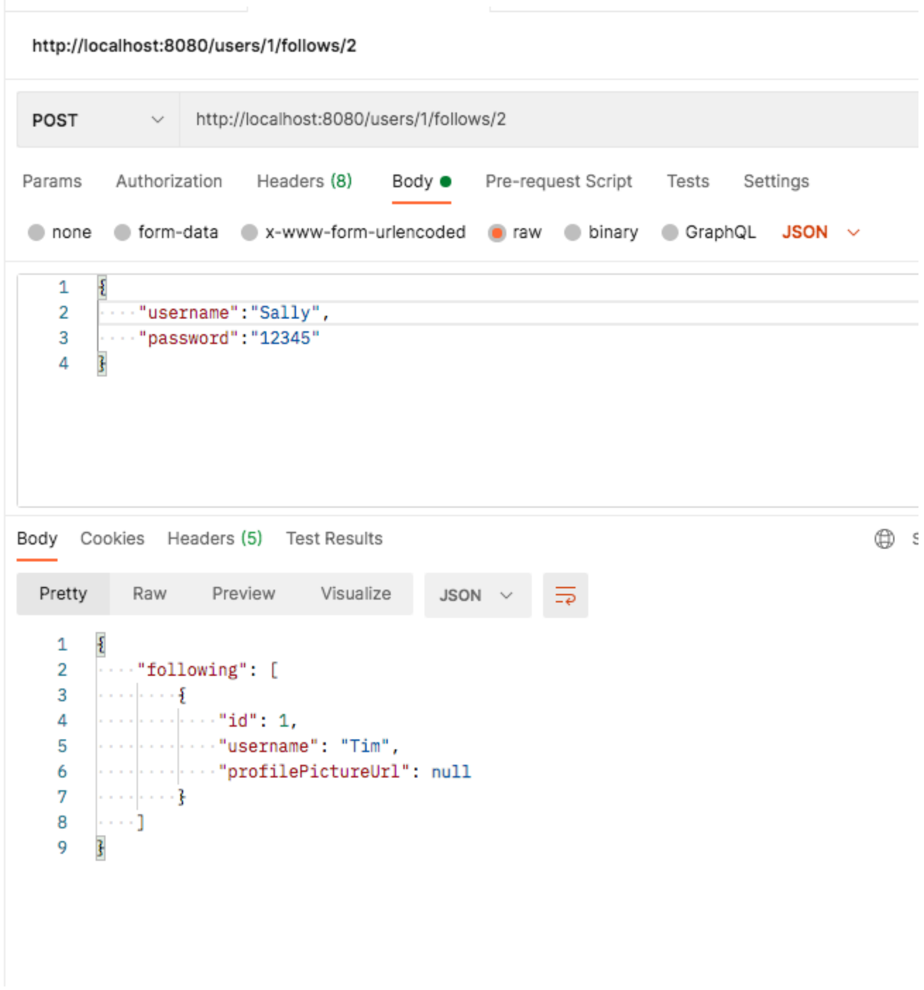
Params Authorization Headers (8) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "username":"Sally",
    "password":"12345"
}
```

Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON

Status: 201 Created

```
{
    "id": 2,
    "username": "Sally",
    "profilePictureUrl": null
}
```



http://localhost:8080/users/1/follows/2

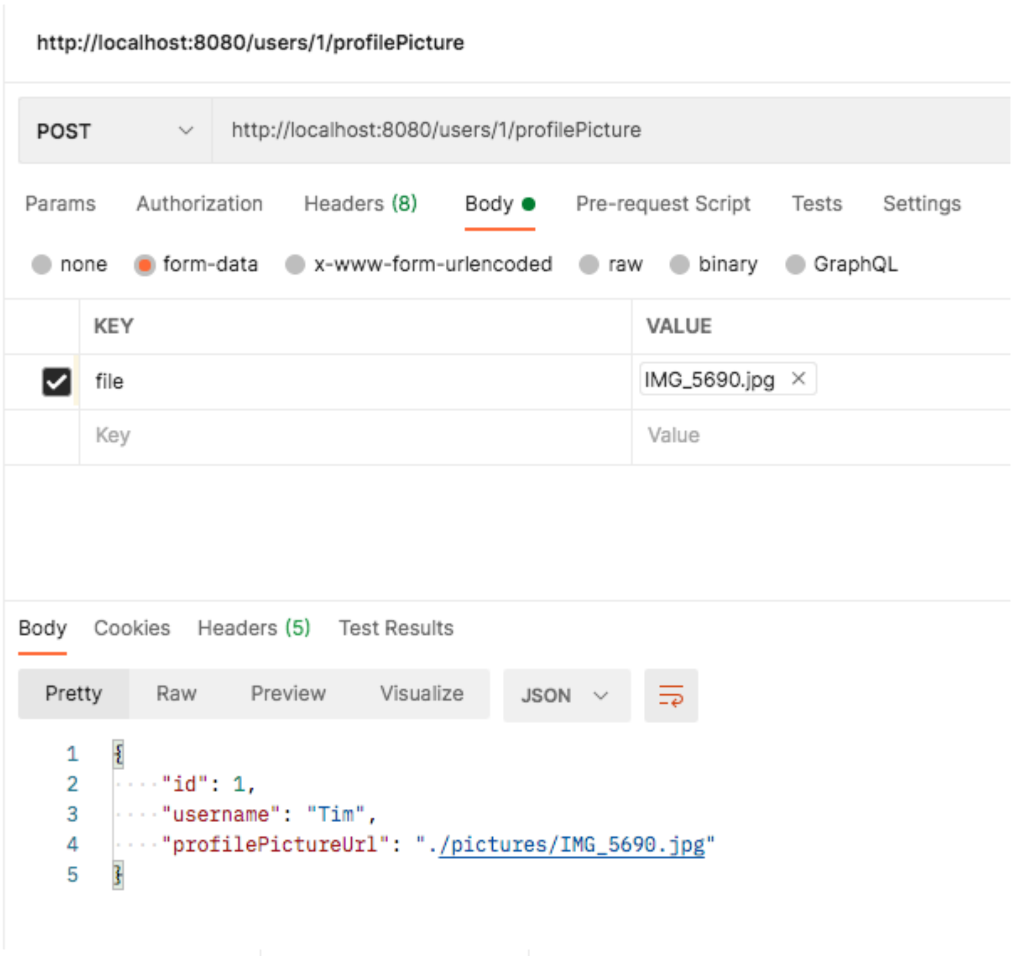
POST http://localhost:8080/users/1/follows/2

Params Authorization Headers (8) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "username":"Sally",
    "password":"12345"
}
```

Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON

```
{
    "following": [
        {
            "id": 1,
            "username": "Tim",
            "profilePictureUrl": null
        }
    ]
}
```



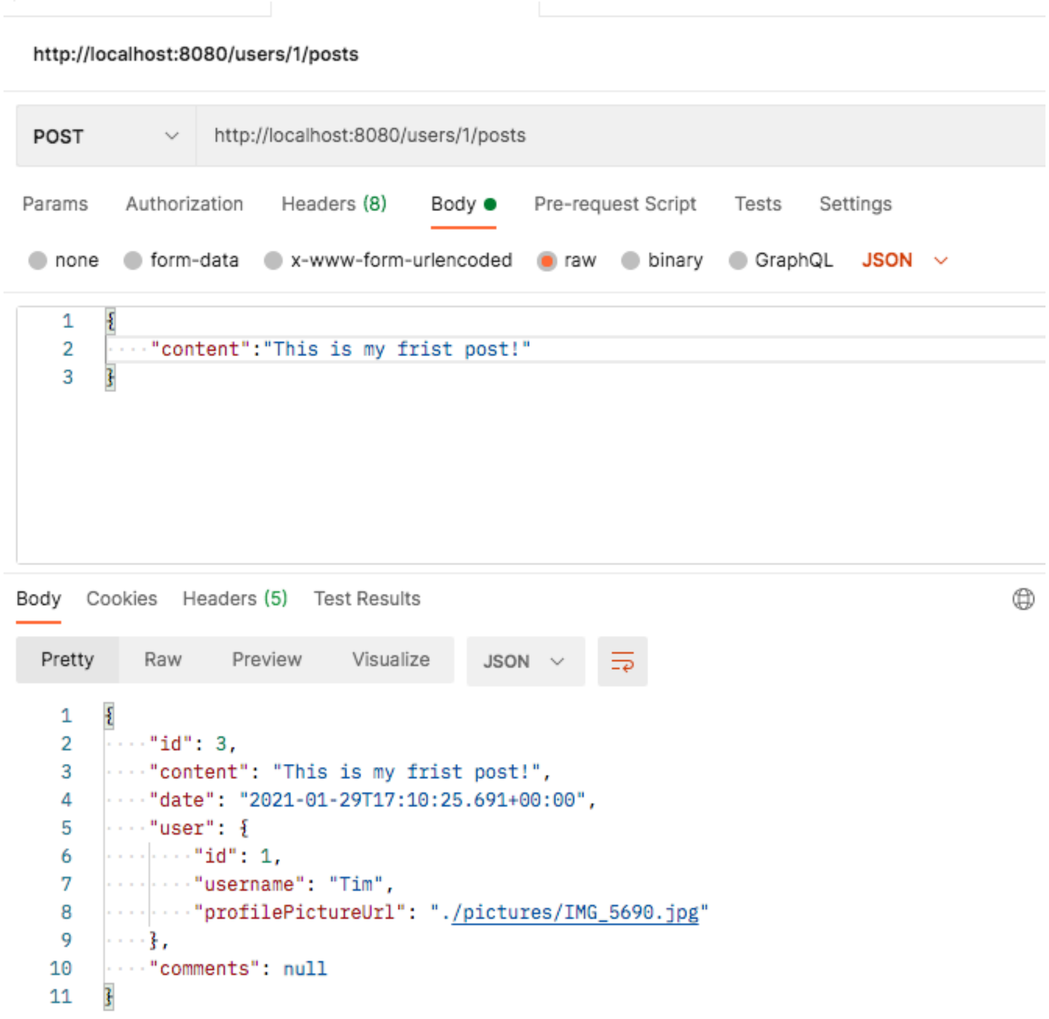
http://localhost:8080/users/1/profilePicture

POST http://localhost:8080/users/1/profilePicture

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

	KEY	VALUE
✓	file	IMG_5690.jpg
	Key	Value

```
{
    "id": 1,
    "username": "Tim",
    "profilePictureUrl": "./pictures/IMG_5690.jpg"
}
```



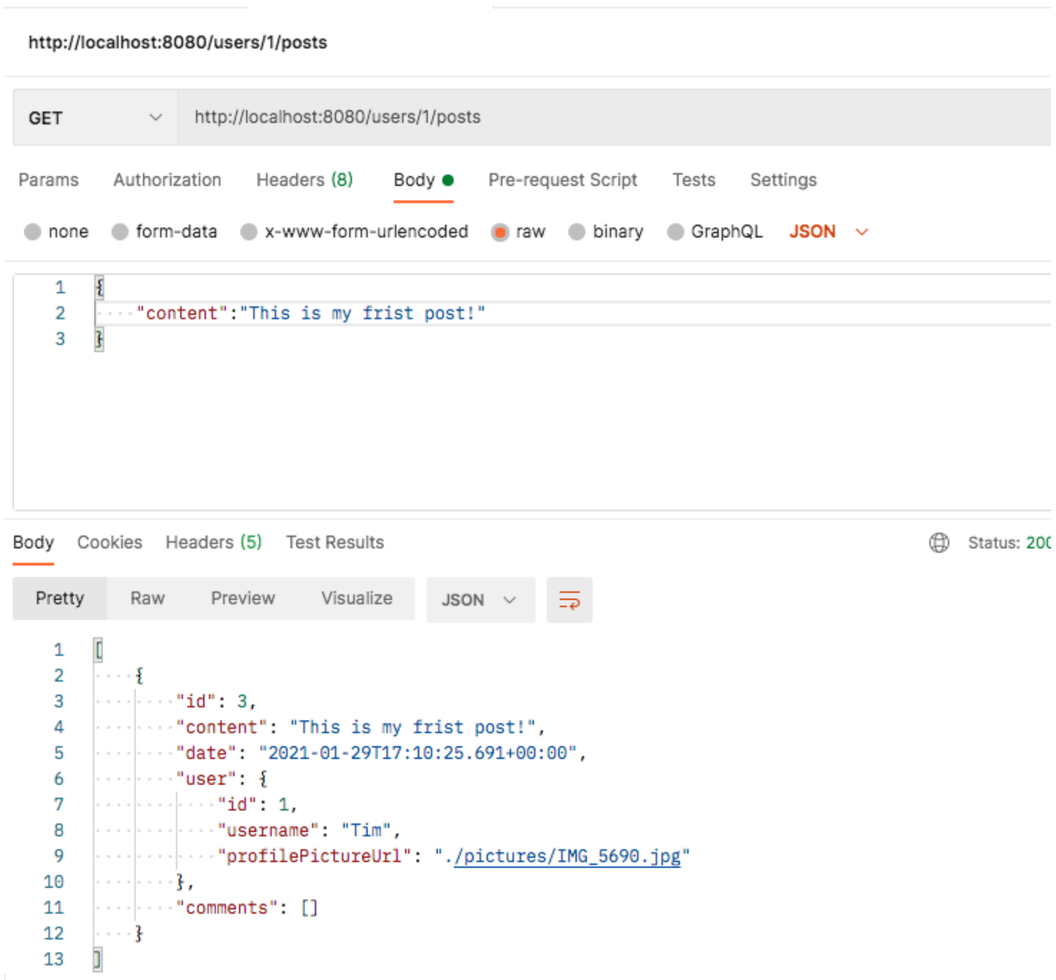
http://localhost:8080/users/1/posts

POST http://localhost:8080/users/1/posts

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

```
{
    "content":"This is my frist post!"
}
```

```
{
    "id": 3,
    "content": "This is my frist post!",
    "date": "2021-01-29T17:10:25.691+00:00",
    "user": {
        "id": 1,
        "username": "Tim",
        "profilePictureUrl": "./pictures/IMG_5690.jpg"
    },
    "comments": null
}
```

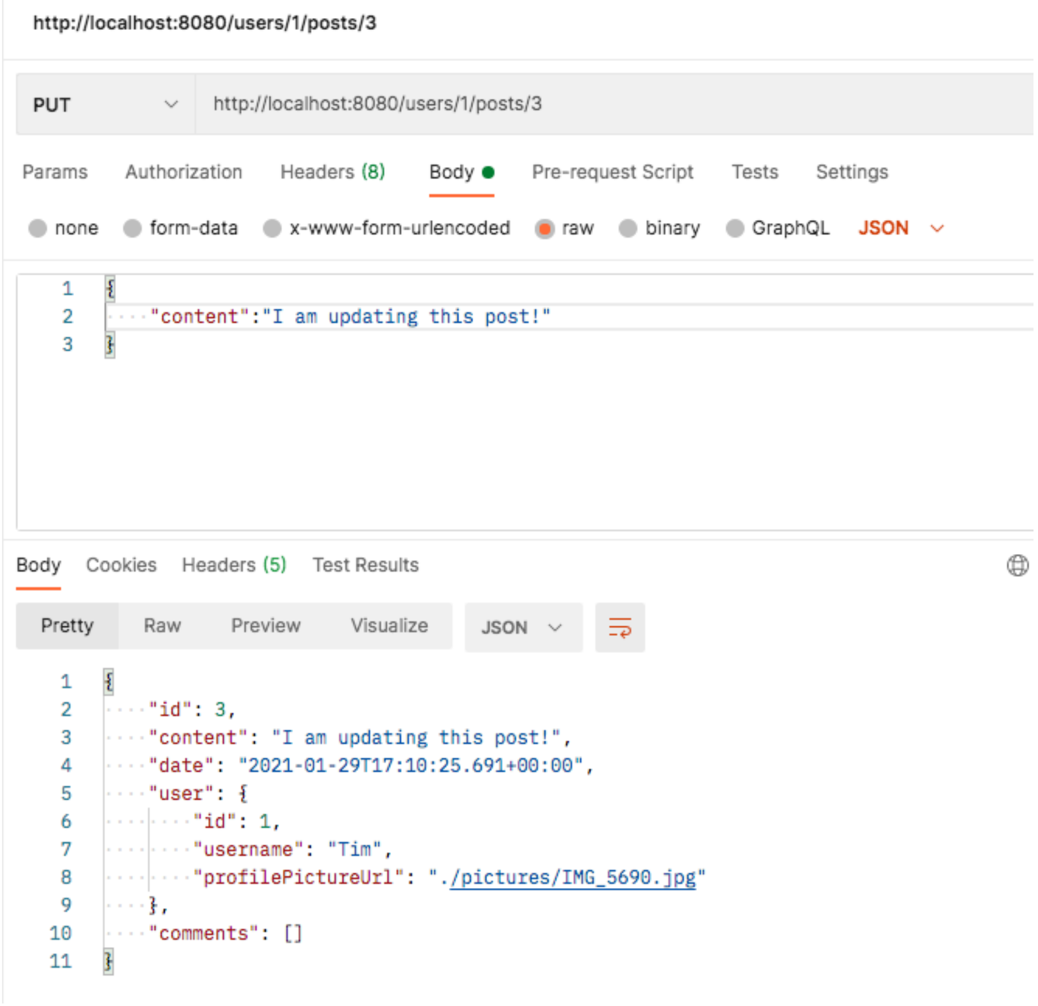


http://localhost:8080/users/1/posts

GET http://localhost:8080/users/1/posts

```
{
    "content":"This is my frist post!"
}
```

```
[
    {
        "id": 3,
        "content": "This is my frist post!",
        "date": "2021-01-29T17:10:25.691+00:00",
        "user": {
            "id": 1,
            "username": "Tim",
            "profilePictureUrl": "./pictures/IMG_5690.jpg"
        },
        "comments": []
    }
]
```

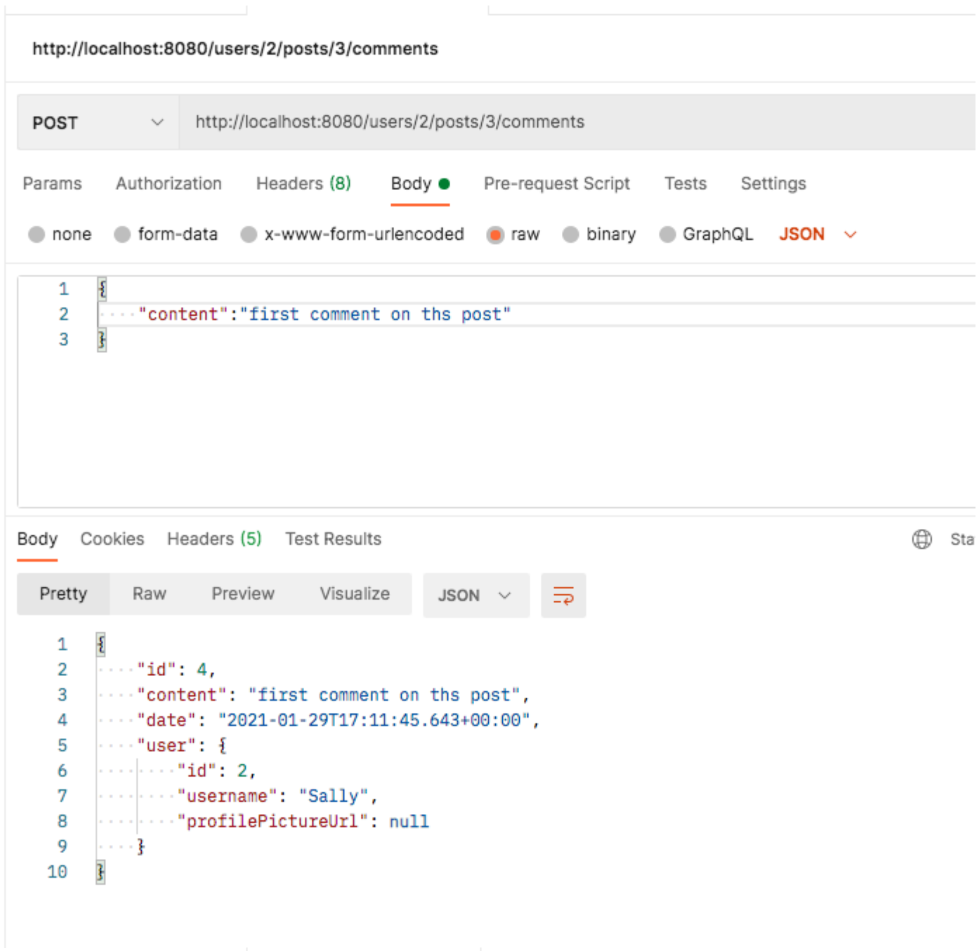


http://localhost:8080/users/1/posts/3

PUT http://localhost:8080/users/1/posts/3

```
{
    "content":"I am updating this post!"
}
```

```
{
    "id": 3,
    "content": "I am updating this post!",
    "date": "2021-01-29T17:10:25.691+00:00",
    "user": {
        "id": 1,
        "username": "Tim",
        "profilePictureUrl": "./pictures/IMG_5690.jpg"
    },
    "comments": []
}
```



http://localhost:8080/users/2/posts/3/comments

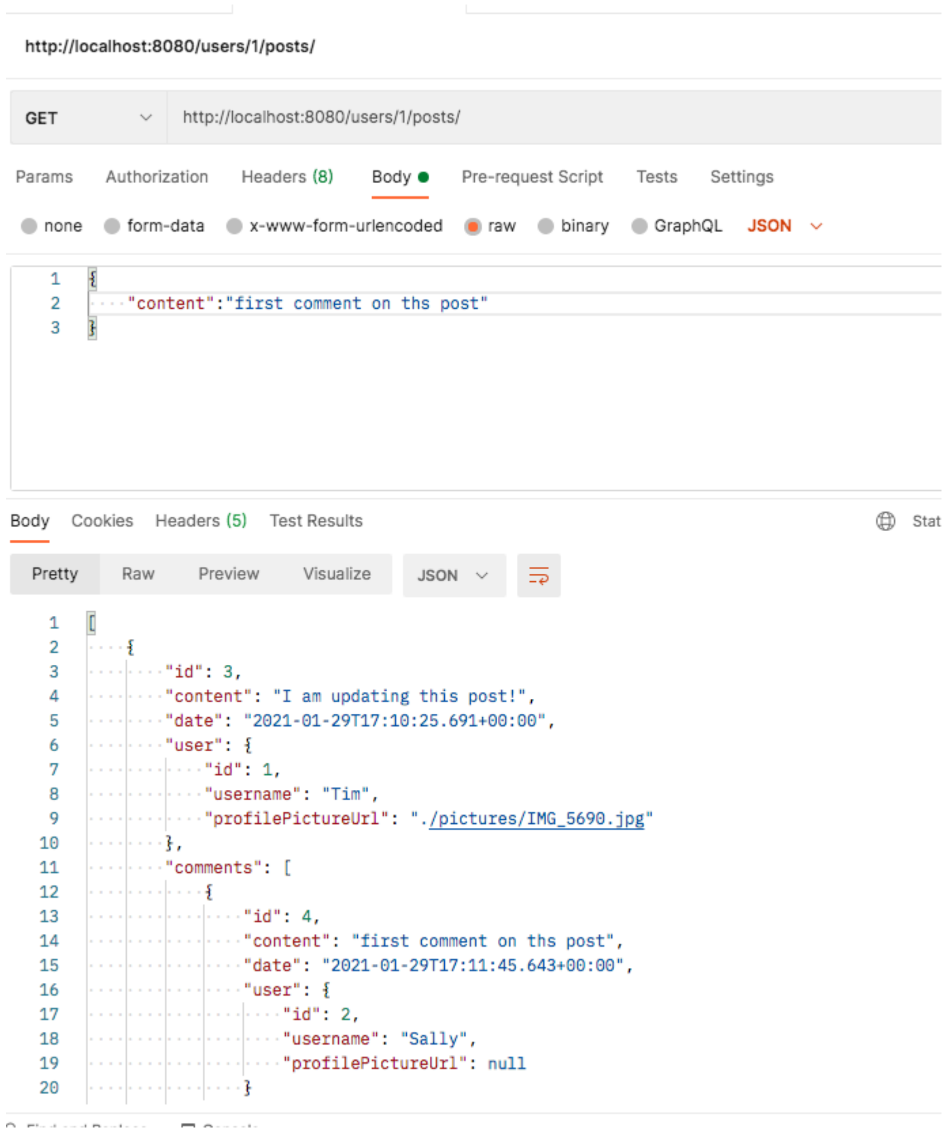
POST http://localhost:8080/users/2/posts/3/comments

Body: raw JSON

```
{
    "content":"first comment on ths post"
}
```

Response Body: Pretty JSON

```
{
    "id": 4,
    "content": "first comment on ths post",
    "date": "2021-01-29T17:11:45.643+00:00",
    "user": {
        "id": 2,
        "username": "Sally",
        "profilePictureUrl": null
    }
}
```



http://localhost:8080/users/1/posts/

GET http://localhost:8080/users/1/posts/

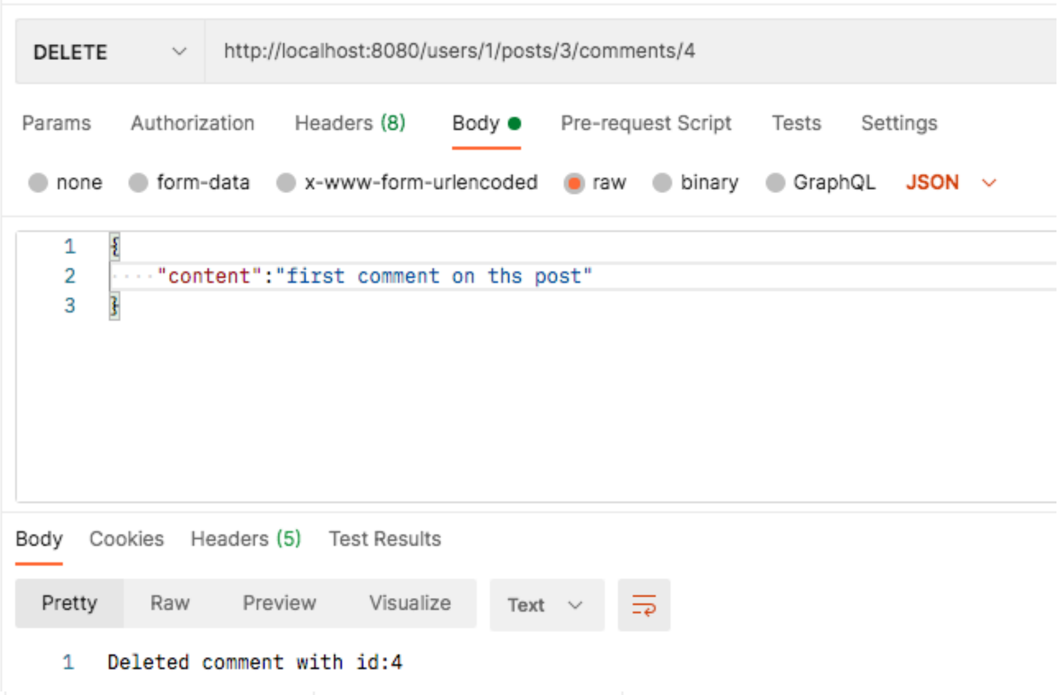
Body: raw JSON

```
{
    "content":"first comment on ths post"
}
```

Response Body: Pretty JSON

```
[
    {
        "id": 3,
        "content": "I am updating this post!",
        "date": "2021-01-29T17:10:25.691+00:00",
        "user": {
            "id": 1,
            "username": "Tim",
            "profilePictureUrl": "./pictures/IMG_5690.jpg"
        },
        "comments": [
            {
                "id": 4,
                "content": "first comment on ths post",
                "date": "2021-01-29T17:11:45.643+00:00",
                "user": {
                    "id": 2,
                    "username": "Sally",
                    "profilePictureUrl": null
                }
```

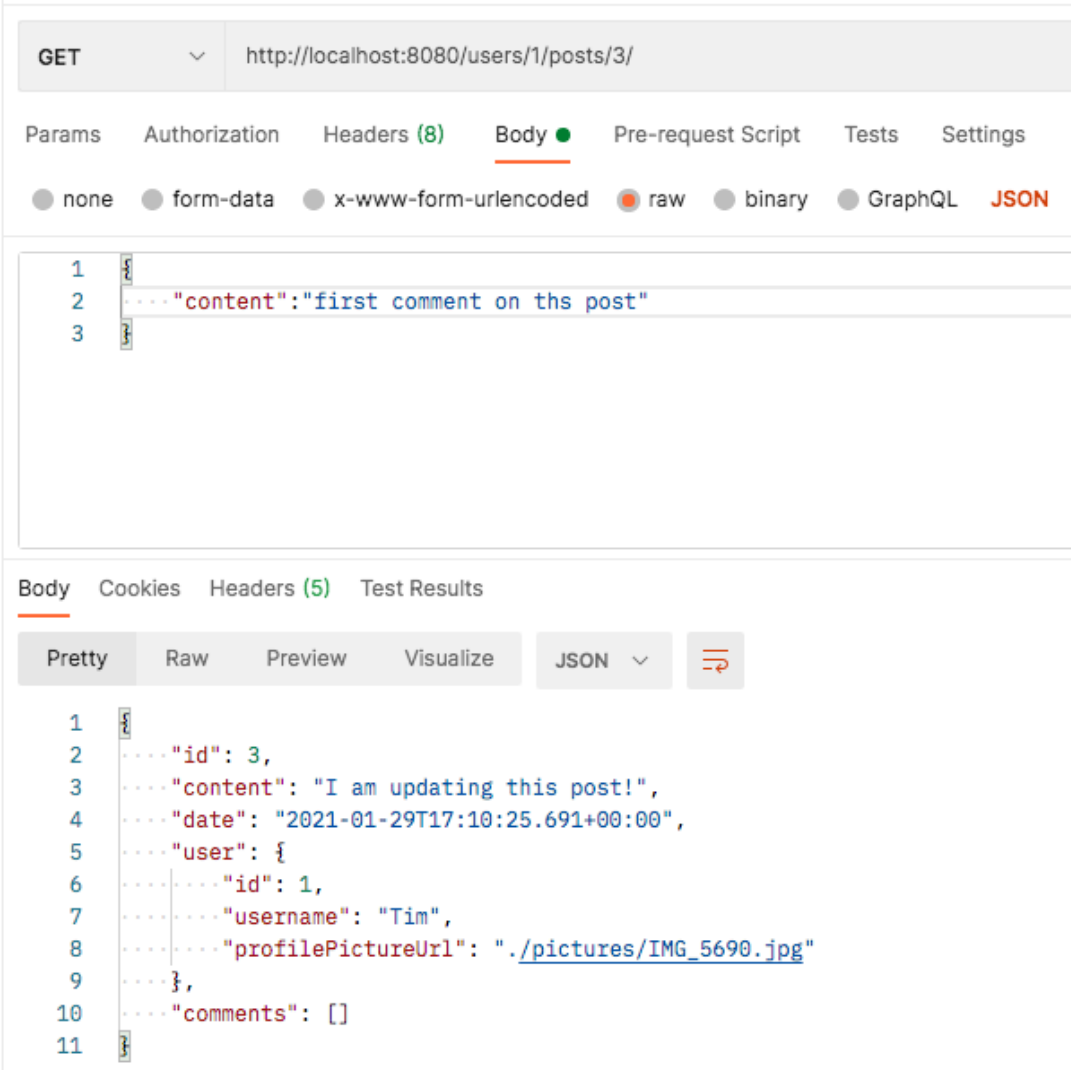
http://localhost:8080/users/1/posts/3/comments/4


DELETE http://localhost:8080/users/1/posts/3/comments/4

```
{
    "content":"first comment on ths post"
}
```

```
Deleted comment with id:4
```

http://localhost:8080/users/1/posts/3/


GET http://localhost:8080/users/1/posts/3/

```
{
    "content":"first comment on ths post"
}
```

```
{
    "id": 3,
    "content": "I am updating this post!",
    "date": "2021-01-29T17:10:25.691+00:00",
    "user": {
        "id": 1,
        "username": "Tim",
        "profilePictureUrl": "./pictures/IMG_5690.jpg"
    },
    "comments": []
}
```

https://github.com/Christinalytle/SocialMediaApi.git