

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=yes&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](#).

```
@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 findById(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 findById(int userId) {

        User findById = userRepo.findById(userId);

        return findById;

    }

    @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 **loadUserByUsername()** 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 exist 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 Filehendlar 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+mul
tipartFile.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.findById(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.findById(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:

GET http://localhost:8080/ POST http://localhost:8080/ + ... No Environment

http://localhost:8080/user/Register Save </>

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

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

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

```
1 {
2   "username": "ARYAN SINGH",
3   "password": "123456789",
4   "email": "aryansingh@gmail.com"
5 }
```

Body Cookies Headers (11) Test Results Status: 200 OK Time: 206 ms Size: 511 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "userId": 20,
3   "username": "ARYAN SINGH",
4   "password": "$2a$10$uL5U0yCo0jYS.LjSDpJUSefydHM629A0t41HBxcg105u1Av3u6xXC",
5   "email": "aryansingh@gmail.com",
6   "roles": "ROLE_NORMAL"
7 }
```

Registration fails:

GET http://localhost:8080/ POST http://localhost:8080/ + ... No Environment

http://localhost:8080/user/Register Save </>

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

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

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

```
1 {
2   "username": "Aryan SINGH",
3   "password": "123456789",
4   "email": "bhojak@gmail.com"
5 }
```

Body Cookies Headers (10) Test Results Status: 400 Bad Request Time: 27 ms Size: 377 B Save Response

Pretty Raw Preview Visualize Text

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

GET http://localhost:8080/

POST http://localhost:8080/

+

...

No Environment

http://localhost:8080/user/Register

Save

POST

http://localhost:8080/user/Register

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Cookies

Beautify

1

2

3

4

5

Body

Cookies

Headers (10)

Test Results

Status: 400 Bad Request

Time: 13 ms

Size: 356 B

Save Response

Pretty

Raw

Preview

Visualize

Text

1

please enter valid email address

Login:

Login Success:

GET http://localhost:8080/

POST http://localhost:8080/

+

...

No Environment

http://localhost:8080/user/Register

Save

POST

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

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Cookies

Beautify

1

2

3

4

Body

Cookies (1)

Headers (12)

Test Results

Status: 200 OK

Time: 274 ms

Size: 450 B

Save Response

Pretty

Raw

Preview

Visualize

Text

1

User signed-in successfully!.

Login fails:

http://localhost:8080/user/Register

Save

POST

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

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324

1325

1326

1327

1328

1329

1330

1331

1332

1333

1334

1335

1336

1337

1338

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

1427

1428

1429

1430

1431

1432

1433

1434

1435

1436

1437

1438

1439

1440

1441

1442

1443

1444

1445

1446

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

1458

1459

1460

1461

1462

1463

1464

1465

1466

1467

1468

1469

1470

1471

1472

1473

1474

Upload posts fails:

GET http://localhost:8080/

POST http://localhost:8080/

+

...

No Environment

http://localhost:8080/user/Register

Save

POST

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

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> file	pancard.pdf			
Key	Value	Description		

Body

Cookies (1)

Headers (10)

Test Results

Status: 500 Internal Server Error

Time: 22 ms

Size: 354 B

Save Response

Pretty

Raw

Preview

Visualize

Text

1 Only jpg is allow

All posts of users:

GET http://localhost:8080/

GET http://localhost:8080/

+

...

No Environment

http://localhost:8080/user/Register

Save

GET

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

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

Body

Cookies (1)

Headers (11)

Test Results

Status: 200 OK

Time: 22 ms

Size: 847 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1 {
2 {
3 "id": 1,
4 "user": {
5 "userId": 1,
6 "username": "manasvi",
7 "password": "\$2a\$10\$d3c/Cv0qxHCnhCv/vf0N.RdITSK8Ez4PK5AhS9upMvLAKSuitbuS",
8 "email": "manasvi@gmail.com",
9 "roles": "ROLE_NORMAL"
10 },
11 "postImg": "http://localhost:8080/image/img3.jpg",
12 "createdDate": "2022-04-23"
13 },
14 {
15 "id": 2,
16 "user": {
17 "userId": 1,
18 "username": "manasvi",
19 "password": "\$2a\$10\$d3c/Cv0qxHCnhCv/vf0N.RdITSK8Ez4PK5AhS9upMvLAKSuitbuS",
20 "email": "manasvi@gmail.com",
21 "roles": "ROLE_NORMAL"
22 }
23 }
24 }

When User have no posts:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/post/allPost/2`. The status is **404 Not Found**, with a response time of 17 ms and a size of 359 B. The response body is displayed in the 'Text' tab as `NO POST`.

Following List:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/user/1/following`. The status is **200 OK**, with a response time of 35 ms and a size of 539 B. The response body is displayed in the 'JSON' tab as a JSON object:

```
1 {
2   "fid": 3,
3   "userId": 1,
4   "user": {
5     "userId": 3,
6     "username": "raj",
7     "password": "$2a$10$kNAjzq7LgBa0Bxoawtv.GuazMU/17E.UWTEYf31jzNeGEeevXzTKW",
8     "email": "raj@gmail.com",
9     "roles": "ROLE_NORMAL"
10  },
11  "following": 1
12 }
13
14 }
```

Followers List:

GET http://localhost:8080/

GET http://localhost:8080/

+

...

No Environment

http://localhost:8080/user/Register

Save

GEThttp://localhost:8080/user/1/follower

Send

ParamsAuthorizationHeaders (9)Body●Pre-request ScriptTestsSettingsCookies

BodyCookies (1)Headers (11)Test Results

Status: 200 OKTime: 20 msSize: 737 BSave Response

PrettyRawPreviewVisualizeJSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

{

"fid": 1,

"userId": 1,

"user": {

"userId": 2,

"username": "monika",

"password": "\$2a\$10\$4thSjc/oLkIcnqzJvLoz3.8Zhj1dewRqvF0HMqdIB2q/30WfYp0pm",

"email": "monika@gmail.com",

"roles": "ROLE_NORMAL"

},

"following": 0

}

,

{

"fid": 3,

"userId": 1,

"user": {

"userId": 3,

"username": "raj",

"password": "\$2a\$10\$kNAjzq7LgBa0Bxoawtv.GuazMU/17E.UWTEYf3ijzNeGEeevXzTKW",

"email": "raj@gmail.com",

"roles": "ROLE_NORMAL"

}

}

User might know people list:

GET http://localhost:8080/

GET http://localhost:8080/

+

...

No Environment

http://localhost:8080/user/Register

Save

GEThttp://localhost:8080/user/1/mightKnown

Send

ParamsAuthorizationHeaders (9)Body●Pre-request ScriptTestsSettingsCookies

BodyCookies (1)Headers (11)Test Results

Status: 200 OKTime: 31 msSize: 539 BSave Response

PrettyRawPreviewVisualizeJSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

{

"fid": 2,

"userId": 2,

"user": {

"userId": 3,

"username": "raj",

"password": "\$2a\$10\$kNAjzq7LgBa0Bxoawtv.GuazMU/17E.UWTEYf3ijzNeGEeevXzTKW",

"email": "raj@gmail.com",

"roles": "ROLE_NORMAL"

},

"following": 1

}

DataBase

User DataBase:

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

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:						
	user_id	email	password	roles	username	is_enabled
1	1	manasvi@gmail.com	\$2a\$10\$rd3c/CvOqxHCnhCv/vf0N.RdITSK8Ez4...	ROLE_NORMAL	manasvi	0
2	2	monika@gmail.com	\$2a\$10\$4thSjc/oLkIcnqzJvLoz3.8Zhj1dewRqv...	ROLE_NORMAL	monika	0
3	3	raj@gmail.com	\$2a\$10\$knAjzq7LgBa0BxoaWtv.GuazMU/7E.U...	ROLE_NORMAL	raj	0
4	4	john@gmail.com	\$2a\$10\$zeE1/Po9g9HJ3v19sXrHBe1Q9epCbcfS...	ROLE_NORMAL	john	0
16	16	bhojakmanu@gmail.com	\$2a\$10\$n5acV6EL5rm8.FSyU2wT0mwV/chIvN...	ROLE_NORMAL	ds22rfdg254f bhojak	0
18	18	manu@gmail.com	\$2a\$10\$v6Tk5805/EIvNHYY2lCHuBRDDHdCjX...	ROLE_NORMAL	ds22r2555fdg254f bhojak	0
20	20	aryansingh@gmail.com	\$2a\$10\$uL5U0yCoOjYS.LjSDpJUSEfydHM629A...	ROLE_NORMAL	ARYAN SINGH	0
*	NULL	NULL	NULL	NULL	NULL	NULL

Post DataBase:

```
1 • 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:

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

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Con

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

Thank You