Project Objective : To createan online quiz portal with multipleREST APIs where users can browse different quizzes, attempt them, and find their scores and standings.

****

Step 1:pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.7.3</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.simplilearn.demo</groupId>
	<artifactId>ExamPortal</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>ExamPortal</name>
	<description>Demo project for Spring Boot</description>
	<properties>
		<java.version>11</java.version>
	</properties>
	<dependencies>
		 <dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-jpa</artifactId>
	  </dependency>
	  <dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	  </dependency>

	  <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-security -->
	  <dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-security</artifactId>

	  </dependency>

	  <!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->
	  <dependency>
		<groupId>io.jsonwebtoken</groupId>
		<artifactId>jjwt</artifactId>
		<version>0.9.0</version>
	  </dependency>
	  <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
	  <dependency>
		<groupId>javax.xml.bind</groupId>
		<artifactId>jaxb-api</artifactId>
		<version>2.3.1</version>
	  </dependency>


	  <dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<scope>runtime</scope>
	  </dependency>
	  <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
      <build>
            <plugins>
                  <plugin>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
                  </plugin>
            </plugins>
      </build>

</project>
**********************************************************************
step 2:application.properties
-----------------------------
#database configuration
spring.datasource.url=jdbc:mysql://localhost:3306/quizapplication
spring.datasource.username=root
spring.datasource.password=Ankit@1998

#jpa configuration
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
server.port=8081

**********************************************************************step
3:ExamPortalApplication.java
--------------------------------
package com.exam;

import com.exam.helper.UserFoundException;
import com.exam.model.Role;
import com.exam.model.User;
import com.exam.model.UserRole;
import com.exam.model.exam.Quiz;
import com.exam.repo.QuizRepository;
import com.exam.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.core.io.Resource;
import org.springframework.core.io.UrlResource;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import
org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import
org.springframework.web.servlet.mvc.method.annotation.MvcUriComponentsBuilder;

import java.io.ByteArrayInputStream;
import java.io.FileInputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
```

```
@SpringBootApplication
public class ExamPortalApplication implements CommandLineRunner {

    @Autowired
    private UserService userService;

    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @Autowired
    public QuizRepository quizRepository;

    public static void main(String[] args) {


        SpringApplication.run(ExamPortalApplication.class, args);


    }

    @Override
    public void run(String... args) throws Exception {
        try {


            System.out.println("starting code");


        } catch (Exception e) {
            e.printStackTrace();


        }


    }


}
**************************************************************************step
4:com.exam.config
-----------------------

jwtAuthenticationEntryPoint.java
--------------------------------
package com.exam.config;

import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {
    @Override
    public void commence(HttpServletRequest request, HttpServletResponse
response, AuthenticationException authException) throws IOException,
ServletException {
```

```
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized :
Server");
    }
}
-----------------------------------------------------------------------
jwtAuthenticationFilter.java
---------------------------
package com.exam.config;

import com.exam.service.impl.UserDetailsServiceImpl;
import io.jsonwebtoken.ExpiredJwtException;
import org.apache.catalina.User;
import org.apache.catalina.security.SecurityConfig;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFi
lter;
import
org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter {


    @Autowired
    private UserDetailsServiceImpl userDetailsService;

    @Autowired
    private JwtUtils jwtUtil;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response, FilterChain filterChain) throws ServletException,
IOException {


        final String requestTokenHeader = request.getHeader("Authorization");
        System.out.println(requestTokenHeader);
        String username = null;
        String jwtToken = null;

        if (requestTokenHeader != null && requestTokenHeader.startsWith("Bearer
")) {
            //yes

            jwtToken = requestTokenHeader.substring(7);

            try {
                username = this.jwtUtil.extractUsername(jwtToken);
            } catch (ExpiredJwtException e) {
                e.printStackTrace();
                System.out.println("jwt token has expired");
```

```
            } catch (Exception e) {
                e.printStackTrace();
                System.out.println("error");
            }

        } else {
            System.out.println("Invalid token , not start with bearer string");
        }

        //validated
        if (username != null &&
SecurityContextHolder.getContext().getAuthentication() == null) {
            final UserDetails userDetails =
this.userDetailsService.loadUserByUsername(username);
            if (this.jwtUtil.validateToken(jwtToken, userDetails)) {
                //token is valid

                UsernamePasswordAuthenticationToken
usernamePasswordAuthentication = new
UsernamePasswordAuthenticationToken(userDetails, null,
userDetails.getAuthorities());
                usernamePasswordAuthentication.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticat
ion);
            }
        } else {
            System.out.println("Token is not valid");
        }

        filterChain.doFilter(request, response);

    }


}
```

---

jwtUtils.java

```
package com.exam.config;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Component
public class JwtUtils {
    private String SECRET_KEY = "examportal";

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
```

```
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver)
{
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

    private Claims extractAllClaims(String token) {
        return
Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token).getBody();
    }

    private Boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, userDetails.getUsername());
    }

    private String createToken(Map<String, Object> claims, String subject) {

        return
Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new
Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 *
60 * 10))
                .signWith(SignatureAlgorithm.HS256, SECRET_KEY).compact();
    }

    public Boolean validateToken(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername()) && !
isTokenExpired(token));
    }
}
-----------------------------------------------------------------------
MySecurityConfig.java
----------------------
package com.exam.config;

import com.exam.service.impl.UserDetailsServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.authentication.builders.Authentic
ationManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobal
MethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
```

```
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFi
lter;

import java.nio.file.Path;
import java.nio.file.Paths;

@EnableWebSecurity
@Configuration
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class MySecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private JwtAuthenticationEntryPoint unauthorizedHandler;

    @Autowired
    private JwtAuthenticationFilter jwtAuthenticationFilter;


    @Autowired
    private UserDetailsServiceImpl userDetailsServiceImpl;

    @Override
    @Bean
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }


    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{

auth.userDetailsService(this.userDetailsServiceImpl).passwordEncoder(passwordEnc
oder());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {


        http
                .csrf()
                .disable()
                .cors()
                .disable()
                .authorizeRequests()
                .antMatchers("/generate-token", "/user/").permitAll()
                .antMatchers(HttpMethod.OPTIONS).permitAll()
                .anyRequest().authenticated()
                .and()
                .exceptionHandling().authenticationEntryPoint(unauthorizedHandle
r)
                .and()
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy
```

```
.STATELESS);

        http.addFilterBefore(jwtAuthenticationFilter,
UsernamePasswordAuthenticationFilter.class);

    }


}
**************************************************************************
step 5:com.exam.controller
--------------------------

AuthenticateController
----------------------
package com.exam.controller;

import com.exam.config.JwtUtils;
import com.exam.helper.UserNotFoundException;
import com.exam.model.JwtRequest;
import com.exam.model.JwtResponse;
import com.exam.model.User;
import com.exam.service.impl.UserDetailsServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Primary;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.DisabledException;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.web.bind.annotation.*;

import java.security.Principal;

@RestController
@CrossOrigin("*")
public class AuthenticateController {


    @Autowired
    private AuthenticationManager authenticationManager;

    @Autowired
    private UserDetailsServiceImpl userDetailsService;

    @Autowired
    private JwtUtils jwtUtils;


    //generate token

    @PostMapping("/generate-token")
    public ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest)
throws Exception {

        try {

            authenticate(jwtRequest.getUsername(), jwtRequest.getPassword());


        } catch (UserNotFoundException e) {
            e.printStackTrace();
```

```
            throw new Exception("User not found ");
        }

        //////////////authenticate

        UserDetails userDetails =
this.userDetailsService.loadUserByUsername(jwtRequest.getUsername());
        String token = this.jwtUtils.generateToken(userDetails);
        return ResponseEntity.ok(new JwtResponse(token));


    }


    private void authenticate(String username, String password) throws Exception
{

        try {

            authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(username, password));

        } catch (DisabledException e) {
            throw new Exception("USER DISABLED " + e.getMessage());
        } catch (BadCredentialsException e) {
            throw new Exception("Invalid Credentials " + e.getMessage());
        }
    }

    //return the details of current user
    @GetMapping("/current-user")
    public User getCurrentUser(Principal principal) {
        return ((User)
this.userDetailsService.loadUserByUsername(principal.getName()));

    }



}
--------------------------------------------------------------------------------
-
CategoryController.java
-----------------------
package com.exam.controller;

import com.exam.model.exam.Category;
import com.exam.service.CategoryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/category")
@CrossOrigin("*")
public class CategoryController {

    @Autowired
    private CategoryService categoryService;

    //add category
    @PostMapping("/")
    public ResponseEntity<Category> addCategory(@RequestBody Category category)
{
```

```
        Category category1 = this.categoryService.addCategory(category);
        return ResponseEntity.ok(category1);
    }

    //get category
    @GetMapping("/{categoryId}")
    public Category getCategory(@PathVariable("categoryId") Long categoryId) {
        return this.categoryService.getCategory(categoryId);
    }

    //get all categories
    @GetMapping("/")
    public ResponseEntity<?> getCategories() {
        return ResponseEntity.ok(this.categoryService.getCategories());
    }

    //update category
    @PutMapping("/")
    public Category updateCategory(@RequestBody Category category) {
        return this.categoryService.updateCategory(category);
    }

    //delete category
    @DeleteMapping("/{categoryId}")
    public void deleteCategory(@PathVariable("categoryId") Long categoryId) {
        this.categoryService.deleteCategory(categoryId);
    }

}
--------------------------------------------------------------------------------
QuestionController.java
------------------------
package com.exam.controller;

import com.exam.model.exam.Question;
import com.exam.model.exam.Quiz;
import com.exam.service.QuestionService;
import com.exam.service.QuizService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.management.Query;
import java.util.*;

@RestController
@CrossOrigin("*")
@RequestMapping("/question")
public class QuestionController {
    @Autowired
    private QuestionService service;

    @Autowired
    private QuizService quizService;

    //add question
    @PostMapping("/")
    public ResponseEntity<Question> add(@RequestBody Question question) {
        return ResponseEntity.ok(this.service.addQuestion(question));
    }

    //update the question
    @PutMapping("/")
    public ResponseEntity<Question> update(@RequestBody Question question) {
```

```
        return ResponseEntity.ok(this.service.updateQuestion(question));
    }

    //get all question of any quiz
    @GetMapping("/quiz/{qid}")
    public ResponseEntity<?> getQuestionsOfQuiz(@PathVariable("qid") Long qid) {
//        Quiz quiz = new Quiz();
//        quiz.setqId(qid);
//        Set<Question> questionsOfQuiz = this.service.getQuestionsOfQuiz(quiz);
//        return ResponseEntity.ok(questionsOfQuiz);

        Quiz quiz = this.quizService.getQuiz(qid);
        Set<Question> questions = quiz.getQuestions();
        List list = new ArrayList(questions);
        if (list.size() > Integer.parseInt(quiz.getNumberOfQuestions())) {
            list = list.subList(0, Integer.parseInt(quiz.getNumberOfQuestions()
+ 1));
        }
        Collections.shuffle(list);
        return ResponseEntity.ok(list);


    }


    @GetMapping("/quiz/all/{qid}")
    public ResponseEntity<?> getQuestionsOfQuizAdmin(@PathVariable("qid") Long
qid) {
        Quiz quiz = new Quiz();
        quiz.setqId(qid);
        Set<Question> questionsOfQuiz = this.service.getQuestionsOfQuiz(quiz);
        return ResponseEntity.ok(questionsOfQuiz);

//        return ResponseEntity.ok(list);


    }
    @GetMapping("/allQuestions/quiz/{quesId}")
    public Question get(@PathVariable("quesId") Long quesId) {
        return this.service.getQuestion(quesId);
    }



    @PostMapping("/allQuestions/Answer/{quesId}")
    public String getQuestionandAnswerandOptions(@PathVariable("quesId") Long
quesId,String SelectedOptions) {
    int result=0;


      if(SelectedOptions.equals(this.service.getQuestion(quesId).getAnswer())) {
            result++;
        return "Correct Answer,your marks is"+result;}
      return  "Wrong Answer, The correct answer is
"+this.service.getQuestion(quesId).getAnswer()+"your marks is "+result;
    }



    @GetMapping("/{quesId}")
    public Set<String> getquestions(@PathVariable("quesId") Long quesId) {
    Set<String> a=new HashSet<>();

    String question=this.service.getQuestion(quesId).getContent();
```

```
    String First=this.service.getQuestion(quesId).getOption1();
    String Second=this.service.getQuestion(quesId).getOption2();
    String Third=this.service.getQuestion(quesId).getOption3();
    String fourth=this.service.getQuestion(quesId).getOption4();
    a.add(question);
    a.add(First);
    a.add(Second);
    a.add(Third);
    a.add(fourth);
      System.out.println(this.service.getQuestion(quesId).getContent());
    System.out.println(this.service.getQuestion(quesId).getOption1());
       System.out.println(this.service.getQuestion(quesId).getOption2());
       System.out.println(this.service.getQuestion(quesId).getOption3());
       System.out.println(this.service.getQuestion(quesId).getOption4());
        return a;
    }
    @GetMapping("/allQuestions/{quesId}")
    public Question getAnswer(@PathVariable("quesId") Long quesId) {
        return this.service.getQuestion(quesId);
    }


    //delete question
    @DeleteMapping("/{quesId}")
    public void delete(@PathVariable("quesId") Long quesId) {
        this.service.deleteQuestion(quesId);
    }


    //eval quiz
    @PostMapping("/eval-quiz")
    public ResponseEntity<?> evalQuiz(@RequestBody List<Question> questions) {
        System.out.println(questions);
        double marksGot = 0;
        int correctAnswers = 0;
        int attempted = 0;
        for (Question q : questions) {
            //single questions
            Question question = this.service.get(q.getQuesId());
            if (question.getAnswer().equals(q.getGivenAnswer())) {
                //correct
                correctAnswers++;

                double marksSingle =
Double.parseDouble(questions.get(0).getQuiz().getMaxMarks()) / questions.size();
                //       this.questions[0].quiz.maxMarks /
this.questions.length;
                marksGot += marksSingle;

            }

            if (q.getGivenAnswer() != null) {
                attempted++;
            }

        }
        ;

        Map<String, Object> map = Map.of("marksGot", marksGot, "correctAnswers",
correctAnswers, "attempted", attempted);
        return ResponseEntity.ok(map);

    }
```

```
}
--------------------------------------------------------------------------------
-
QuizController.java
--------------------
package com.exam.controller;

import com.exam.model.exam.Category;
import com.exam.model.exam.Quiz;
import com.exam.service.QuizService;
import org.apache.coyote.Response;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.RequestEntity;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Map;

@RestController
@CrossOrigin("*")
@RequestMapping("/quiz")
public class QuizController {
    @Autowired
    private QuizService quizService;

    //add quiz service
    @PostMapping("/")
    public ResponseEntity<Quiz> add(@RequestBody Quiz quiz) {
        return ResponseEntity.ok(this.quizService.addQuiz(quiz));
    }

    //update quiz

    @PutMapping("/")
    public ResponseEntity<Quiz> update(@RequestBody Quiz quiz) {
        return ResponseEntity.ok(this.quizService.updateQuiz(quiz));
    }

    //get quiz
    @GetMapping("/")
    public ResponseEntity<?> quizzes() {
        return ResponseEntity.ok(this.quizService.getQuizzes());
    }

    //get single quiz
    @GetMapping("/{qid}")
    public Quiz quiz(@PathVariable("qid") Long qid) {
        return this.quizService.getQuiz(qid);
    }

    //delete the quiz
    @DeleteMapping("/{qid}")
    public void delete(@PathVariable("qid") Long qid) {
        this.quizService.deleteQuiz(qid);
    }

    @GetMapping("/category/{cid}")
    public List<Quiz> getQuizzesOfCategory(@PathVariable("cid") Long cid) {
        Category category = new Category();
        category.setCid(cid);
        return this.quizService.getQuizzesOfCategory(category);
    }
```

```
    //get active quizzes
    @GetMapping("/active")
    public List<Quiz> getActiveQuizzes() {
        return this.quizService.getActiveQuizzes();
    }

    //get active quizzes of category
    @GetMapping("/category/active/{cid}")
    public List<Quiz> getActiveQuizzes(@PathVariable("cid") Long cid) {
        Category category = new Category();
        category.setCid(cid);
        return this.quizService.getActiveQuizzesOfCategory(category);
    }




}
--------------------------------------------------------------------------------
-
UserController.java
---------------------
package com.exam.controller;

import com.exam.helper.UserFoundException;
import com.exam.helper.UserNotFoundException;
import com.exam.model.Role;
import com.exam.model.User;
import com.exam.model.UserRole;
import com.exam.service.UserService;
import org.apache.coyote.Response;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.*;

import java.util.HashSet;
import java.util.Set;

@RestController
@RequestMapping("/user")
@CrossOrigin("*")
public class UserController {

    @Autowired
    private UserService userService;

    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    //creating user
    @PostMapping("/")
    public User createUser(@RequestBody User user) throws Exception {


        user.setProfile(user.getProfile());
        //encoding password with bcryptpasswordencoder

        user.setPassword(this.bCryptPasswordEncoder.encode(user.getPassword()));

        Set<UserRole> roles = new HashSet<>();

        Role role = new Role();
        role.setRoleId(46L);
```

```
        role.setRoleName("User");

        UserRole userRole = new UserRole();
        userRole.setUser(user);
        userRole.setRole(role);

        roles.add(userRole);


        return this.userService.createUser(user, roles);

    }

    @GetMapping("/{username}")
    public User getUser(@PathVariable("username") String username) {
        return this.userService.getUser(username);
    }

    //delete the user by id
    @DeleteMapping("/{userId}")
    public void deleteUser(@PathVariable("userId") Long userId) {
        this.userService.deleteUser(userId);
    }


    //update api


    @ExceptionHandler(UserFoundException.class)
    public ResponseEntity<?> exceptionHandler(UserFoundException ex) {
        return ResponseEntity.ok(ex.getMessage());
    }


}
*********************************************************************************
*step 6:com.exam.helper
-----------------------
UserFoundException
------------------
package com.exam.helper;

public class UserFoundException  extends  Exception{

    public UserFoundException() {
        super("User with this Username is already there in DB !! try with
another one");
    }

    public UserFoundException(String msg)
    {
        super(msg);
    }
}
-------------------------------------------------------------------------------
-
UserNotFoundException.java
--------------------------
package com.exam.helper;

public class UserNotFoundException extends Exception {

    public UserNotFoundException() {
        super("User with this username not found in database !!");
```

```
    }

    public UserNotFoundException(String msg) {
        super(msg);
    }
}
*********************************************************************************
*
step 7:com.exam.model
----------------------
Authority.java
---------------
package com.exam.model;

import org.springframework.security.core.GrantedAuthority;

public class Authority implements GrantedAuthority {

    private String authority;

    public Authority(String authority) {
        this.authority = authority;
    }

    @Override
    public String getAuthority() {
        return this.authority;
    }
}
---------------------------------------------------------------------------------
-
jwtRequest.java
----------------
package com.exam.model;

public class JwtRequest {
    String username;
    String password;

    public JwtRequest() {
    }

    public JwtRequest(String username, String password) {

        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

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

    public String getPassword() {
        return password;
    }

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

```
--------------------------------------------------------------------------------
-
jwtResponse.java
-----------------
package com.exam.model;

public class JwtResponse {
    String token;

    public JwtResponse(String token) {
        this.token = token;
    }

    public JwtResponse() {
    }

    public String getToken() {
        return token;
    }

    public void setToken(String token) {
        this.token = token;
    }
}
--------------------------------------------------------------------------------
-
user.java
---------
package com.exam.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import java.util.Collection;
import java.util.HashSet;
import java.util.Set;

@Entity
@Table(name = "users")
public class User implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String username;
    private String password;
    private String firstName;
    private String lastName;
    private String email;
    private String phone;
    private boolean enabled = true;
    private String profile;

    //user many roles

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, mappedBy =
"user")
    @JsonIgnore
    private Set<UserRole> userRoles = new HashSet<>();


    public User() {
```

```
    }

    public Set<UserRole> getUserRoles() {
        return userRoles;
    }

    public void setUserRoles(Set<UserRole> userRoles) {
        this.userRoles = userRoles;
    }

    public User(Long id, String username, String password, String firstName,
String lastName, String email, String phone, boolean enabled, String profile) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.phone = phone;
        this.enabled = enabled;
        this.profile = profile;
    }

    public String getProfile() {
        return profile;
    }

    public void setProfile(String profile) {
        this.profile = profile;
    }

    public Long getId() {
        return id;
    }

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

    public String getUsername() {
        return username;
    }

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

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

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

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

    @Override
```

```
    public Collection<? extends GrantedAuthority> getAuthorities() {

        Set<Authority> set = new HashSet<>();
        this.userRoles.forEach(userRole -> {
            set.add(new Authority(userRole.getRole().getRoleName()));
        });


        return set;
    }

    public String getPassword() {
        return password;
    }

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

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public boolean isEnabled() {
        return enabled;
    }

    public void setEnabled(boolean enabled) {
        this.enabled = enabled;
    }
}
--------------------------------------------------------------------------------
-
userRole.java
--------------
package com.exam.model;
```

```
import javax.persistence.*;

@Entity
public class UserRole {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long userRoleId;

    //user
    @ManyToOne(fetch = FetchType.EAGER)
    private User user;

    @ManyToOne
    private Role role;

    public UserRole() {
    }

    public Long getUserRoleId() {
        return userRoleId;
    }

    public void setUserRoleId(Long userRoleId) {
        this.userRoleId = userRoleId;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    }
}
*********************************************************************************
*
step 8:com.exam.model.exam
--------------------------
category.java
-------------
package com.exam.model.exam;

import com.fasterxml.jackson.annotation.JsonIgnore;

import javax.persistence.*;
import java.util.LinkedHashSet;
import java.util.Set;

@Entity
@Table(name = "category")
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long cid;
```

```
    private String title;

    private String description;

    @OneToMany(mappedBy = "category", cascade = CascadeType.ALL)
    @JsonIgnore
    private Set<Quiz> quizzes = new LinkedHashSet<>();

    public Category() {
    }

    public Category(String title, String description) {
        this.title = title;
        this.description = description;
    }

    public Long getCid() {
        return cid;
    }

    public void setCid(Long cid) {
        this.cid = cid;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
--------------------------------------------------------------------------------
-
Question.java
--------------
package com.exam.model.exam;

import com.fasterxml.jackson.annotation.JsonIgnore;

import javax.persistence.*;

@Entity
public class Question {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long quesId;
    @Column(length = 5000)
    private String content;

    private String image;

    public Question(Long quesId, String content, String image, String
selectedOptions, String option1, String option2,
```

```
                String option3, String option4, String answer, String
givenAnswer, Quiz quiz) {
            super();
            this.quesId = quesId;
            this.content = content;
            this.image = image;

            this.option1 = option1;
            this.option2 = option2;
            this.option3 = option3;
            this.option4 = option4;
            this.answer = answer;
            this.givenAnswer = givenAnswer;
            this.quiz = quiz;
      }



      private String option1;
    private String option2;
    private String option3;
    private String option4;


    private String answer;

    @Transient
    private  String givenAnswer;

    @ManyToOne(fetch = FetchType.EAGER)
    private Quiz quiz;

    public Question() {
    }

    public Long getQuesId() {
        return quesId;
    }

    public void setQuesId(Long quesId) {
        this.quesId = quesId;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    public String getOption1() {
        return option1;
    }

    public void setOption1(String option1) {
```

```
        this.option1 = option1;
    }

    public String getOption2() {
        return option2;
    }

    public void setOption2(String option2) {
        this.option2 = option2;
    }

    public String getOption3() {
        return option3;
    }

    public void setOption3(String option3) {
        this.option3 = option3;
    }

    public String getOption4() {
        return option4;
    }

    public void setOption4(String option4) {
        this.option4 = option4;
    }

    public String getAnswer() {
        return answer;
    }

    public void setAnswer(String answer) {
        this.answer = answer;
    }

    public Quiz getQuiz() {
        return quiz;
    }

    public void setQuiz(Quiz quiz) {
        this.quiz = quiz;
    }

    public String getGivenAnswer() {
        return givenAnswer;
    }

    public void setGivenAnswer(String givenAnswer) {
        this.givenAnswer = givenAnswer;
    }
}
--------------------------------------------------------------------------------
-
Quiz.java
----------
package com.exam.model.exam;

import com.fasterxml.jackson.annotation.JsonIgnore;

import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
```

```
public class Quiz {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long qId;


    private String title;

    @Column(length = 5000)
    private String description;

    private String maxMarks;

    private String numberOfQuestions;

    private boolean active = false;
    //add..

    @ManyToOne(fetch = FetchType.EAGER)
    private Category category;

    @OneToMany(mappedBy = "quiz", fetch = FetchType.LAZY, cascade =
CascadeType.ALL)
    @JsonIgnore
    private Set<Question> questions = new HashSet<>();


    public Quiz() {
    }

    public Long getqId() {
        return qId;
    }

    public void setqId(Long qId) {
        this.qId = qId;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getMaxMarks() {
        return maxMarks;
    }

    public void setMaxMarks(String maxMarks) {
        this.maxMarks = maxMarks;
    }

    public String getNumberOfQuestions() {
```

```
        return numberOfQuestions;
    }

    public void setNumberOfQuestions(String numberOfQuestions) {
        this.numberOfQuestions = numberOfQuestions;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    public Category getCategory() {
        return category;
    }

    public void setCategory(Category category) {
        this.category = category;
    }

    public Set<Question> getQuestions() {
        return questions;
    }

    public void setQuestions(Set<Question> questions) {
        this.questions = questions;
    }
}
*********************************************************************************
*
step 9:com.exam.repo
---------------------
CategoryRepository
--------------------
package com.exam.repo;

import com.exam.model.exam.Category;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CategoryRepository extends JpaRepository<Category, Long> {
}
-------------------------------------------------------------------------------
-
QuestionRepository
------------------
package com.exam.repo;

import com.exam.model.exam.Question;
import com.exam.model.exam.Quiz;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Set;

public interface QuestionRepository extends JpaRepository<Question, Long> {
    Set<Question> findByQuiz(Quiz quiz);
}
-------------------------------------------------------------------------------
-QuizRepository.java
--------------------
package com.exam.repo;
```

```
import com.exam.model.exam.Category;
import com.exam.model.exam.Quiz;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import java.util.List;

public interface QuizRepository extends JpaRepository<Quiz, Long> {
    public List<Quiz> findBycategory(Category category);

    public List<Quiz> findByActive(Boolean b);

    public List<Quiz> findByCategoryAndActive(Category c, Boolean b);
}
----------------------------------------------------------------------------
-RoleRepository.java
--------------------
package com.exam.repo;

import com.exam.model.Role;
import org.springframework.data.jpa.repository.JpaRepository;

public interface RoleRepository extends JpaRepository<Role,Long> {
}
----------------------------------------------------------------------------
-UserRepository.java
--------------------
package com.exam.repo;

import com.exam.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

    public User findByUsername(String username);
}
****************************************************************************
*step 10: com.exam.service
--------------------------
CategoryService.java
--------------------
package com.exam.repo;

import com.exam.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

    public User findByUsername(String username);
}
----------------------------------------------------------------------------
-QuestionService.java
---------------------
package com.exam.service;

import com.exam.model.exam.Question;
import com.exam.model.exam.Quiz;

import java.util.Set;

public interface QuestionService {

    public Question addQuestion(Question question);
```

```
    public Question updateQuestion(Question question);

    public Set<Question> getQuestions();

    public Question getQuestion(Long questionId);

    public Set<Question> getQuestionsOfQuiz(Quiz quiz);

    public void deleteQuestion(Long quesId);

    public Question get(Long questionsId);

}
--------------------------------------------------------------------------------
-QuizService.java
-----------------
package com.exam.service;

import com.exam.model.exam.Category;
import com.exam.model.exam.Quiz;
import org.springframework.http.ResponseEntity;

import java.util.List;
import java.util.Set;

public interface QuizService {

    public Quiz addQuiz(Quiz quiz);

    public Quiz updateQuiz(Quiz quiz);

    public Set<Quiz> getQuizzes();

    public Quiz getQuiz(Long quizId);

    public void deleteQuiz(Long quizId);


    public List<Quiz> getQuizzesOfCategory(Category category);

    public List<Quiz> getActiveQuizzes();

    public List<Quiz> getActiveQuizzesOfCategory(Category c);
}
--------------------------------------------------------------------------------
-
UserService.java
----------------
package com.exam.service;

import com.exam.model.User;
import com.exam.model.UserRole;

import java.util.Set;

public interface UserService {

    //creating user
    public User createUser(User user, Set<UserRole> userRoles) throws Exception;

    //get user by username
    public User getUser(String username);

    //delete user by id
```

```
    public void deleteUser(Long userId);
}
*****************************************************************************
*step 11:com.exam.service.impl
-----------------------------
CategoryServiceImpl.java
------------------------
package com.exam.service.impl;

import com.exam.model.exam.Category;
import com.exam.repo.CategoryRepository;
import com.exam.service.CategoryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.LinkedHashSet;
import java.util.Set;

@Service
public class CategoryServiceImpl implements CategoryService {

    @Autowired
    private CategoryRepository categoryRepository;

    @Override
    public Category addCategory(Category category) {
        return this.categoryRepository.save(category);
    }

    @Override
    public Category updateCategory(Category category) {
        return this.categoryRepository.save(category);
    }

    @Override
    public Set<Category> getCategories() {
        return new LinkedHashSet<>(this.categoryRepository.findAll());
    }

    @Override
    public Category getCategory(Long categoryId) {
        return this.categoryRepository.findById(categoryId).get();
    }

    @Override
    public void deleteCategory(Long categoryId) {
        Category category = new Category();
        category.setCid(categoryId);
        this.categoryRepository.delete(category);
    }
}
-------------------------------------------------------------------------------
-
QuestionServiceImpl.java
------------------------
package com.exam.service.impl;

import com.exam.model.exam.Question;
import com.exam.model.exam.Quiz;
import com.exam.repo.QuestionRepository;
import com.exam.service.QuestionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import java.util.HashSet;
import java.util.Set;

@Service
public class QuestionServiceImpl implements QuestionService {

    @Autowired
    private QuestionRepository questionRepository;

    @Override
    public Question addQuestion(Question question) {
        return this.questionRepository.save(question);
    }

    @Override
    public Question updateQuestion(Question question) {
        return this.questionRepository.save(question);
    }

    @Override
    public Set<Question> getQuestions() {
        return new HashSet<>(this.questionRepository.findAll());
    }

    @Override
    public Question getQuestion(Long questionId) {
        return this.questionRepository.findById(questionId).get();
    }

    @Override
    public Set<Question> getQuestionsOfQuiz(Quiz quiz) {
        return this.questionRepository.findByQuiz(quiz);
    }

    @Override
    public void deleteQuestion(Long quesId) {
        Question question = new Question();
        question.setQuesId(quesId);
        this.questionRepository.delete(question);
    }

    @Override
    public Question get(Long questionsId) {
       return this.questionRepository.getOne(questionsId);
    }
}
--------------------------------------------------------------------------------
-
QuizServiceImpl.java
---------------------
package com.exam.service.impl;

import com.exam.model.exam.Category;
import com.exam.model.exam.Quiz;
import com.exam.repo.QuizRepository;
import com.exam.service.QuizService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
```

```
@Service
@Transactional
public class QuizServiceImpl implements QuizService {
    @Autowired
    private QuizRepository quizRepository;

    @Override
    public Quiz addQuiz(Quiz quiz) {
        return this.quizRepository.save(quiz);
    }

    @Override
    public Quiz updateQuiz(Quiz quiz) {
        return this.quizRepository.save(quiz);
    }

    @Override
    public Set<Quiz> getQuizzes() {
        return new HashSet<>(this.quizRepository.findAll());
    }

    @Override
    public Quiz getQuiz(Long quizId) {
        return this.quizRepository.findById(quizId).get();
    }

    @Override
    public void deleteQuiz(Long quizId) {
        this.quizRepository.deleteById(quizId);
    }

    @Override
    public List<Quiz> getQuizzesOfCategory(Category category) {
        return this.quizRepository.findBycategory(category);
    }


    //get active quizzes

    @Override
    public List<Quiz> getActiveQuizzes() {
        return this.quizRepository.findByActive(true);
    }

    @Override
    public List<Quiz> getActiveQuizzesOfCategory(Category c) {
        return this.quizRepository.findByCategoryAndActive(c, true);
    }

}
--------------------------------------------------------------------------------
-
UserDetailsServiceImpl.java
----------------------------
package com.exam.service.impl;

import com.exam.model.User;
import com.exam.repo.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
```

```
@Service
public class UserDetailsServiceImpl implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {

        User user = this.userRepository.findByUsername(username);
        if (user == null) {
            System.out.println("User not found");
            throw new UsernameNotFoundException("No user found !!");
        }

        return user;
    }
}
-------------------------------------------------------------------------------
-
UserServiceImpl.java
---------------------
package com.exam.service.impl;

import com.exam.helper.UserFoundException;
import com.exam.helper.UserNotFoundException;
import com.exam.model.Role;
import com.exam.model.User;
import com.exam.model.UserRole;
import com.exam.repo.RoleRepository;
import com.exam.repo.UserRepository;
import com.exam.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;

import java.util.Set;

@Service
public class UserServiceImpl implements UserService {


    @Autowired
    private UserRepository userRepository;

    @Autowired
    private RoleRepository roleRepository;

    //creating user
    @Override
    public User createUser(User user, Set<UserRole> userRoles) throws Exception
{


        User local = this.userRepository.findByUsername(user.getUsername());
        if (local != null) {
            System.out.println("User is already there !!");
            throw new UserFoundException();
        } else {
            //user create
            for (UserRole ur : userRoles) {
                roleRepository.save(ur.getRole());
            }
```

```
            user.getUserRoles().addAll(userRoles);
            local = this.userRepository.save(user);

        }

        return local;
    }

    //getting user by username
    @Override
    public User getUser(String username) {
        return this.userRepository.findByUsername(username);
    }

    @Override
    public void deleteUser(Long userId) {
        this.userRepository.deleteById(userId);
    }


}
*****************************************************************************
*
```