**Backend :-**

**1.AuthenticateController :-**

package com.exam.controller;

import java.security.Principal;

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

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.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.exam.config.JwtUtil;

import com.exam.helper.UserNotFoundException;

import com.exam.model.JwtRequest;

import com.exam.model.JwtResponse;

import com.exam.model.User;

@RestController

@CrossOrigin("\*")

public class AuthenticateController {

    @Autowired

    private AuthenticationManager authenticationManager;

    @Autowired

    private UserDetailsService userDetailsService;

    @Autowired

    private JwtUtil jwtUtil;

    // 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.jwtUtil.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());

        }

    }

    @GetMapping("/current-user")

    public User getCurrentUser(Principal *principal*) {

        return (User) this.userDetailsService.loadUserByUsername(principal.getName());

    }

}

**2. CategoryController :-**

package com.exam.controller;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.exam.model.exam.Category;

import com.exam.service.CategoryService;

@RestController

@RequestMapping("/category")

@CrossOrigin("\*")

public class CategoryController {

    @Autowired

    private CategoryService categoryService;

    //add category

    @PostMapping("/")

    public ResponseEntity<?> addCategory(@RequestBody Category *category*){

        Category category1=this.categoryService.addCategory(category);

        return ResponseEntity.ok(category1);

    }

    // getcategory

    @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);

    }

}

**3.QuestionController**

package com.exam.controller;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Collections;

import java.util.List;

import java.util.Map;

import java.util.Set;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.exam.model.exam.Question;

import com.exam.model.exam.Quiz;

import com.exam.service.QuestionService;

import com.exam.service.QuizService;

@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.upateQuestion(question));

    }

    // get all 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 <Question> list=new ArrayList(questions);

        if(list.size() > Integer.parseInt(quiz.getNumberOfQuestions())) {

            list=list.subList(0, Integer.parseInt(quiz.getNumberOfQuestions()+1));

        }

        list.forEach((q)->{

            q.setAnswer("");

        });

        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);

    }

    //get single question

    @GetMapping("/{quesId}")

    public Question get(@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 = (double) 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();

                           marksGot += marksSingle;

           }

           if (q.getGivenAnswer()!=null ){

               attempted++;

               //

           }

        };

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

        return ResponseEntity.ok(map);

    }

}

**4.QuizController :-**

package com.exam.controller;

import java.util.List;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.exam.model.exam.Category;

import com.exam.model.exam.Quiz;

import com.exam.service.QuizService;

@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 quizzes

    @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> getActiveQuizzesOfCategory(@PathVariable("cid") Long *cid*){

    Category category=new Category();

        category.setCid(cid);

        return this.quizService.getActiveQuizzesofCategory(category);

}

}

**5.UserController :-**

package com.exam.controller;

import java.util.HashSet;

import java.util.Set;

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.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.exam.helper.UserFoundException;

import com.exam.model.Role;

import com.exam.model.User;

import com.exam.model.UserRole;

import com.exam.service.UserService;

@RestController

@RequestMapping("/user")

@CrossOrigin("\*")

public class UserController {

    @Autowired

    private UserService userService;

    @Autowired

    private BCryptPasswordEncoder bCryptPasswordEncoder;

    // creating user                     // we can return direct  responsentity oruser

    @PostMapping("/")

    public User createUser(@RequestBody User *user*) throws Exception {

        user.setProfile("defualt.png");

        // encoding pssword with bcryptpasswordencoder

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

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

        Role role=new Role();

        role.setRoleId(45L);

        role.setRoleName("NORMAL");                    // we applied restriction jo bhi ayega vo normal user add hoga

        UserRole userRole=new UserRole();

        userRole.setUser(user);

        userRole.setRole(role);

        roles.add(userRole);

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

    }

    // get the user

    @GetMapping("/{username}")

    public User getUser(@PathVariable("username") String *username*) {

        return this.userService.getUser(username);

    }

    //delete user by id

    @DeleteMapping("/{userId}")

    public void deleteUser(@PathVariable("userId") Long *userId*) {

        this.userService.deleteUser(userId);

    }

    // update the user

    @ExceptionHandler(UserFoundException.class)

    public ResponseEntity<?> exceptionHandler(UserFoundException *ex*){

        return ResponseEntity.ok(ex.getMessage());

    }

}

**Frontend :-**

1. **Admin :-**

import { Component, OnInit } from '@angular/core';

import { MatSnackBar } from '@angular/material/snack-bar';

import { CategoryService } from 'src/app/services/category.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-add-category',

  templateUrl: './add-category.component.html',

  styleUrls: ['./add-category.component.css']

})

export class AddCategoryComponent implements OnInit {

  category={

    title:'',

    description:'',

  }

  constructor(private *\_category*:CategoryService,private *\_snak*:MatSnackBar) { }

  ngOnInit(): void {

  }

  formSubmit(){

    if(this.category.title.trim()=='' || this.category.title==null){

      this.\_snak.open("Title Required !!",'',{

           duration:3000,

      })

      return;

    }

    // all done

    this.\_category.addCategory(this.category).subscribe(

      (*data*:any)=>{

        this.category.title='';

        this.category.description='';

        Swal.fire("Sucess !!",'Category is added suceesfully','success')

      },

        (*error*)=>{

          console.log(error)

          Swal.fire('Error !!','Server error !!','error')

        }

    );

  }

}

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

import { QuestionService } from 'src/app/services/question.service';

import Swal from 'sweetalert2';

import \* as ClassicEditor from '@ckeditor/ckeditor5-build-classic';

@Component({

  selector: 'app-add-question',

  templateUrl: './add-question.component.html',

  styleUrls: ['./add-question.component.css'],

})

export class AddQuestionComponent implements OnInit {

  public Editor = ClassicEditor;

  qid: any;

  QTitle: any;

  question={

    quiz:{

     qId:Number,

    },

    content:'',

    option1:'',

    option2:'',

    option3:'',

    option4:'',

    answer:'',

  }

  constructor(

    private *\_route*:ActivatedRoute,

    private *\_question*:QuestionService

  ) { }

  ngOnInit(): void {

    this.qid=this.\_route.snapshot.params['qid'];

    this.QTitle=this.\_route.snapshot.params['title'];

    this.question.quiz.qId=this.qid;

  }

  formSubmit(){

    if(this.question.content.trim()=='' || this.question.content==null){

      return ;

    }

    if(this.question.option1.trim()=='' || this.question.option1==null){

      return ;

    }

    if(this.question.option2.trim()=='' || this.question.option2==null){

      return;

    }

    if(this.question.answer.trim()=='' || this.question.answer==null){

      return;

    }

      // form submit

      this.\_question.addQuestion(this.question).subscribe(

        (*data*:any)=>{

              Swal.fire('Success','Question Added','success')

              this.question.content=''

              this.question.option1=''

              this.question.option2=''

              this.question.option3=''

              this.question.option4=''

              this.question.answer=''

        },

        (*error*)=>{

          Swal.fire('Error','Error in adding question','error')

        }

      );

  }

}

import { Component, OnInit } from '@angular/core';

import { QuizService } from 'src/app/services/quiz.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-view-quizzes',

  templateUrl: './view-quizzes.component.html',

  styleUrls: ['./view-quizzes.component.css']

})

export class ViewQuizzesComponent implements OnInit {

 // quizzes=[

    // {

    //   qId:23,

    //   title:'Basic Java Quiz',

    //   description:'java is best programming language for learneres',

    //   maxMarks:'50',

    //   numberOfQuestions:'20',

    //   active:'',

    //   category:{

    //     title:'Programming'

    //   }

    // },

 // ]

 quizzes:any

  constructor(private *\_quiz*:QuizService) { }

  ngOnInit(): void {

    this.\_quiz.quizzes().subscribe(

      (*data*:any)=>{

        this.quizzes=data;

        console.log(this.quizzes)

      },

      (*error*)=>{

        console.log(error)

        Swal.fire('Error !','Error in loading data !','error')

      }

    );

  }

   // delete quiz

   deleteQuiz(*qId*:any){

    Swal.fire({

      icon:'info',

      title:'Are you sure ?',

      confirmButtonText: 'Delete',

      showCancelButton:true,

    }).then((*result*)=>{

      if(result.isConfirmed){

        // delete

        this.\_quiz.deleteQuiz(qId).subscribe(

          (*data*:any)=>{

           //  list of quizzes se filter out kijiye,qId!=qId : ye hua to this.quizzes to ismai add kro

            this.quizzes=this.quizzes.filter((*quiz*:any)=>quiz.qId!=qId);

            Swal.fire('Sucess','Quiz deleted','success')

          },

            (*error*)=>{

              console.log(error)

              Swal.fire('Error','Error in deleting quiz','error')

            });

      }

    }

    )

     }

     // update quiz

     // quiz ki old id send, send new data

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Login :-**
2. import { Component, OnInit } from '@angular/core';
3. import { MatSnackBar } from '@angular/material/snack-bar';
4. import { Router } from '@angular/router';
5. import { LoginService } from 'src/app/services/login.service';
6. @Component({
7. selector: 'app-login',
8. templateUrl: './login.component.html',
9. styleUrls: ['./login.component.css']
10. })
11. export class LoginComponent implements OnInit {
12. loginData = {
13. username: '',
14. password: ''
15. };
16. constructor(private *snack*: MatSnackBar, private *login*: LoginService,private *router*:Router) { }
18. ngOnInit(): void {
19. }
20. formSubmit() {
21. console.log('login btn clicked')
22. if (this.loginData.username.trim() == '' ||
23. this.loginData.username == null) {
24. this.snack.open('username is required !!', '', {
25. duration: 3000
26. })
27. return;
28. }
29. if (this.loginData.password.trim() == '' ||
30. this.loginData.password == null) {
31. this.snack.open('password is required !!', '', {
32. duration: 3000
33. })
34. return;
35. }
36. // // request to server to generate to generate token
37. this.login.generateToken(this.loginData).subscribe(
38. (*data*: any) => {
39. console.log("success");
40. console.log(data)
41. // login..
42. this.login.loginUser(data.token);            // set token in localstorage
43. this.login.getCurrentUser().subscribe(
44. (*user*:any)=>{
45. this.login.setUser(user);            // set the userdetails in localstoarge
46. console.log(user)
47. // redirect ...ADMIN : admin dashboard
48. // reidrect ...NORMAL : normal-dashboard
50. if(this.login.getUserRole()=="ADMIN"){
51. // admin dashboard
52. // window.location.href='/admin'
53. this.router.navigate(['admin'])
54. this.login.loginStatusSubject.next(true)
55. }else if(this.login.getUserRole()=="NORMAL"){
56. // normal user dashboard
57. // window.location.href='/user-dashboard/';
58. this.router.navigate(['user-dashboard/0'])
59. this.login.loginStatusSubject.next(true)
60. }else{
61. this.login.logout();
62. }
63. }
64. )
66. },
67. (*error*) => {
68. console.log('Error !');
69. console.log(error);
70. this.snack.open("Inavalid Details !! Try again",'',{
71. duration:3000
72. })
73. }
75. );
76. }

79. }

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**3. SignUp :-**

import { Component, OnInit } from '@angular/core';

import { MatSnackBar } from '@angular/material/snack-bar';

import { UserService } from 'src/app/services/user.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-signup',

  templateUrl: './signup.component.html',

  styleUrls: ['./signup.component.css']

})

export class SignupComponent implements OnInit {

  constructor(private *userService*:UserService,private *snack*:MatSnackBar) { }

//1. you can make oject by using inrerface or class or 2. direct create object

  public user={

    username:'',

    password:'',

    firstName:'',

    lastName:'',

    email:'',

    phone:''

  };

  ngOnInit(): void {}

formSubmit(){

  // alert("submit")

  console.log(this.user);

  if(this.user.username=='' || this.user.username == null){

    // alert('user is required !!')

    this.snack.open("Username is required !!",'',{

      duration:3000,

      // verticalPosition:'top',

      // horizontalPosition:'right'

    });

    return;

  }

    // validate

  //add :userservice

  this.userService.addUser(this.user).subscribe(

    (*data*:any)=>{

      //success

      console.log(data)

      // alert('success')

      Swal.fire('Successfully done !!','user id is'+ data.id, 'success');

    },

    (*error*:any)=>{

      // error

      console.log(error)

      // alert('something went wrong')

      this.snack.open(error.error.text,'',{

        duration:3000

      })

    }

  )

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**4. User :-**

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { QuizService } from 'src/app/services/quiz.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-instruction',

  templateUrl: './instruction.component.html',

  styleUrls: ['./instruction.component.css']

})

export class InstructionComponent implements OnInit {

  qid: any

  quiz: any

  constructor(

    private *\_route*:ActivatedRoute,

    private *\_quiz*:QuizService,

    private *\_router*:Router

  ) { }

  ngOnInit(): void {

    this.qid=this.\_route.snapshot.params['qid']

    // console.log(this.qid)

    this.\_quiz.getQuiz(this.qid).subscribe(

      (*data*)=>{

        this.quiz=data

        console.log(data)

      },

*error*=>{

        console.log(error)

      }

    )

  }

  //

  startQuiz(){

    // copy form sweetalert2

    Swal.fire({

      title: 'Do you want to start the Quiz?',

      showCancelButton: true,

      confirmButtonText: 'Start',

      denyButtonText: `Don't save`,

      icon:'info'

    }).then((*result*) => {

      /\* Read more about isConfirmed, isDenied below \*/

      if (result.isConfirmed) {

        this.\_router.navigate(['/start/'+this.quiz.qId])

      } else if (result.isDenied) {

        Swal.fire('Changes are not saved', '', 'info')

      }

    })

  }

}

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

import { QuizService } from 'src/app/services/quiz.service';

@Component({

  selector: 'app-load-quiz',

  templateUrl: './load-quiz.component.html',

  styleUrls: ['./load-quiz.component.css']

})

export class LoadQuizComponent implements OnInit {

  catId: any

  quizzes: any;

  constructor(

    private *\_route*: ActivatedRoute,

    private *\_quiz*: QuizService) { }

  ngOnInit(): void {

    this.\_route.params.subscribe((*params*)=>{

      this.catId=params['catId'];

      if (this.catId == 0) {

        console.log("Load all the quiz")

        this.\_quiz.getActiveQuizzes().subscribe(

          (*data*: any) => {

            this.quizzes = data

            console.log(this.quizzes)

          },

          (*error*) => {

            console.log(error)

        }

        );

      } else {

        console.log("Load specific quiz");

        this.\_quiz.getActiveQuizzesOfCategory(this.catId).subscribe(

          (*data*:any)=>

          {

            this.quizzes=data;

            console.log(this.quizzes);

          },

*error*=>{

            // alert("error in loading data")

            // this.quizzes=null

          }

        )

      }

    })

  }

}

import { Component, OnInit } from '@angular/core';

import { MatSnackBar } from '@angular/material/snack-bar';

import { CategoryService } from 'src/app/services/category.service';

@Component({

  selector: 'app-sidebar-user',

  templateUrl: './sidebar.component.html',

  styleUrls: ['./sidebar.component.css']

})

export class SidebarComponent implements OnInit {

  categories:any

  constructor(private *\_cat*:CategoryService,private *\_snak*:MatSnackBar) { }

  ngOnInit(): void {

    this.\_cat.categories().subscribe(

      (*data*:any)=>{

        this.categories=data

      },

*error*=>{

        this.\_snak.open("Error in loading categories from server",'',{

          duration:3000,

        })

      }

    )

  }

}

import { LocationStrategy } from '@angular/common';

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

import { QuestionService } from 'src/app/services/question.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-start',

  templateUrl: './start.component.html',

  styleUrls: ['./start.component.css']

})

export class StartComponent implements OnInit {

  qid: any;

  questions: any;

  marksGot = 0

  correctAnswer = 0

  attempted = 0

  isSubmit = false;

  timer: any

  constructor(

    private *locationSt*: LocationStrategy,

    private *\_route*: ActivatedRoute,

    private *\_question*: QuestionService

  ) { }

  ngOnInit(): void {

    this.preventBackButton();

    this.qid = this.\_route.snapshot.params['qid']

    console.log(this.qid)

    this.loadQuestions();

  }

  loadQuestions() {

    this.\_question.getQuestionsOfQuizForTest(this.qid).subscribe(

      (*data*: any) => {

        // console.log(data)

        this.questions = data;

        this.timer = this.questions.length \* 2 \* 60;

        // this.questions.forEach((q: any) => {

        //   q['givenAnswer'] = ''

        // });

        console.log(this.questions)

        this.startTimer();

      },

*error* => {

        console.log(error)

        Swal.fire("Error", "Error in loading questions", 'error')

      }

    )

  }

  preventBackButton() {

    history.pushState(null, '', location.href);

    this.locationSt.onPopState(() => {

      history.pushState(null, '', location.href);

    })

  }

  submitQuiz() {

    Swal.fire({

      title: 'Do you want to submit the Quiz?',

      showCancelButton: true,

      confirmButtonText: 'submit',

      icon: 'info'

    }).then((*e*) => {

      if (e.isConfirmed) {

        // calculation

        this.evalQuiz();

      }

    })

  }

  startTimer() {

    let t = window.setInterval(() => {

      //code

      if (this.timer <= 0) {

        this.evalQuiz();

        clearInterval(t)

      } else {

        this.timer--;

      }

    }, 1000)

  }

  getFormattedTime() {

    let mm = Math.floor(this.timer / 60);

    let ss = this.timer - mm \* 60;

    return `${mm} min: ${ss} sec`

  }

  // function for submitting quiz for timer or Submit quiz button

  evalQuiz() {

    // calculation

//------------------

    // call the server to evaluate the quiz

    this.\_question.evalQuiz(this.questions).subscribe(

      (*data*: any)=>{

        console.log(data)

        this.marksGot=parseFloat(Number(data.marksGot).toFixed(2))

        this.correctAnswer=data.correctAnswer

        this.attempted=data.attempted

        this.isSubmit = true

      },

      (*error*)=>{

        console.log(error)

      }

    )

//------------------------------

    // this.isSubmit = true,

    //   console.log(this.questions)

    // this.questions.forEach((q: any) => {

    //   if (q.givenAnswer == q.answer) {

    //     this.correctAnswer++;

    //     let marksSingle = this.questions[0].quiz.maxMarks / this.questions.length

    //     this.marksGot += marksSingle

    //   }

    //   if (q.givenAnswer.trim() != '') {

    //     this.attempted++

    //   }

    // });

    // console.log("Correct Answers :" + this.correctAnswer)

    // console.log("Marks Got :" + this.marksGot)

    // console.log("attempted :" + this.attempted)

  }

  printPage(){

    window.print();

  }

}