**Online Quiz Portal Using REST APIs**

This document contains section for

* Sprint planning and Task completion
* Core concept used in project
* Flow of the Application
* Demonstrating the product capabilities, appearance and user interaction
* Conclusion

The code for this project is hosted at <https://github.com/Prateekdu/Phase-1-Practice-Project.git>

The project is developed by **Prateek Dubey**.

**Sprints planning and Task completion**

The project is planned to be completed in 1 sprint .Task assumed to be completed in the sprint are:

* Creating the flow of the application
* Initialization git repository to track changes as development progresses.
* Writing the java program to fulfill the requirements of the project.
* Testing the java program with different kinds of user input.
* Pushing code to GitHub.
* Creating this specification document highlighting application capabilities, appearance ,and user interactions.

**Concepts used in project**

* MYSQL
* Spring Boot
* Hibernate
* Rest API
* Tool 🡪 Postman

**Flow of the Application**

Admin

Start

User

Login API by using username & password

Create Account

NO

YES

NO

YES

Authenticated

Register

Re-Enter the correct details

Add question in quiz

Create Quiz

User attempt the quiz using quizid

Enter quiz id in quiz

Quiz is available for user

Finally quiz is created

Admin can see the result

Result

**Code**

**OnlineQuizApplication.java**

package com;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

@SpringBootApplication(scanBasePackages = "com")

@EntityScan(basePackages = "com.bean")

public class OnlineQuizApplication {

public static void main(String[] args) {

SpringApplication.run(OnlineQuizApplication.class, args);

System.out.println("Quiz Server Started");

}

}

**Com.controller**

**Quiz.Controller.java**

package com.controller;

import java.util.List;

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

import org.springframework.http.MediaType;

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

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

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

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

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

import com.bean.AdminLogin;

import com.bean.Participants;

import com.bean.Questions;

import com.bean.Quiz;

import com.bean.UserQuiz;

import com.service.QuizService;

@RestController

public class QuizController {

@Autowired

QuizService quizService;

// http://localhost:8080/checkAdmin/

@RequestMapping(value = "checkAdmin/{emailid}/{password}",method=RequestMethod.POST)

public String checkAdminLogin(@PathVariable("emailid") String emailid,@PathVariable("password") String password) {

//List<AdminLogin> listAdmin = quizservice.checkadminDetails();

if(emailid.equals("Prateek@quiz.in") && password.equals("12345"))

{ return "Admin logged in successfully";

}

else {

return "Admin Not Found";

}

}

// http://localhost:8080/signUp/

@RequestMapping(value = "signUp",method=RequestMethod.POST,consumes = MediaType.APPLICATION\_JSON\_VALUE)

public String signUp(@RequestBody Participants pt) {

return quizService.storeParticipant(pt);

}

// http://localhost:8080/checkParticipants/

@RequestMapping(value = "checkParticipants/{emailid}/{password}",method=RequestMethod.POST)

public String checkParticipantsLogin(@PathVariable("emailid") String emailid,@PathVariable("password") String password) {

List<Participants> listOfParticipants = quizService.getAllParticipants();

Participants s = listOfParticipants.get(0);

if(s.equals(emailid) && s.equals(password)) {

return "Paticipant logged in successfully";

}

else {

return "Paticipant Not Found";

}

}

//http://localhost:8080/getAllParticipants/

@RequestMapping(value = "getAllParticipants",method=RequestMethod.GET,consumes = MediaType.APPLICATION\_JSON\_VALUE)

public List<Participants> getAllParticipants() {

return quizService.getAllParticipants();

}

//http://localhost:8080/addQuestion/

@RequestMapping(value = "addQuestion",method=RequestMethod.POST,consumes = MediaType.APPLICATION\_JSON\_VALUE)

public String addQuestion(@RequestBody Questions q) {

return quizService.storeQuestion(q);

}

// http://localhost:8080/findQuestionsById/1

@RequestMapping(value = "findQuestionsById/{qid}",method = RequestMethod.GET,produces = MediaType.APPLICATION\_JSON\_VALUE)

public Questions findQuestionsByIdUsingPathParam(@PathVariable("qid") int id) {

return quizService.findQuestions(id);

}

//check this

//http://localhost:8080/createQuiz/

@RequestMapping(value = "createQuiz",method=RequestMethod.POST,consumes = MediaType.APPLICATION\_JSON\_VALUE)

public String createQuiz(@RequestBody Quiz qu) {

return quizService.createQuiz(qu);

}

//http://localhost:8080/takeQuiz/

@RequestMapping(value = "takeQuiz",method=RequestMethod.POST,consumes = MediaType.APPLICATION\_JSON\_VALUE)

public String takeQuizQuiz(@RequestBody UserQuiz uq) {

return quizService.takeQuiz(uq);

}

//http://localhost:8080/checkResult/

@RequestMapping(value = "checkResult",method=RequestMethod.GET,consumes = MediaType.APPLICATION\_JSON\_VALUE)

public String checkResult() {

return quizService.checkResult();

}

}

**Com.bean**

**AdminLogin.java**

package com.bean;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class AdminLogin {

@Id

private String emailid;

private String password;

public String getEmailid() {

return emailid;

}

public void setEmailid(String emailid) {

this.emailid = emailid;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

@Override

public String toString() {

return "AdminLogin [emailid=" + emailid + ", password=" + password + "]";

}

}

**Participants.java**

package com.bean;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="participants")

public class Participants {

@Id

private String emailid;

private String name;

private String password;

private String phoneno;

public String getEmailid() {

return emailid;

}

public void setEmailid(String emailid) {

this.emailid = emailid;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getPhoneno() {

return phoneno;

}

public void setPhoneno(String phoneno) {

this.phoneno = phoneno;

}

@Override

public String toString() {

return "Participants [emailid=" + emailid + ", name=" + name + ", password=" + password + ", phoneno=" + phoneno

+ "]";

}

}

**Questions.java**

package com.bean;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Questions {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private int qid;

private String question;

private String a;

private String b;

private String c;

private String d;

private String correctanswer;

public String getQuestion() {

return question;

}

public void setQuestion(String question) {

this.question = question;

}

public String getA() {

return a;

}

public void setA(String a) {

this.a = a;

}

public String getB() {

return b;

}

public void setB(String b) {

this.b = b;

}

public String getC() {

return c;

}

public void setC(String c) {

this.c = c;

}

public String getD() {

return d;

}

public void setD(String d) {

this.d = d;

}

public String getCorrectanswer() {

return correctanswer;

}

public void setCorrectanswer(String correctanswer) {

this.correctanswer = correctanswer;

}

public int getQid() {

return qid;

}

@Override

public String toString() {

return "Questions [qid=" + qid + ", question=" + question + ", a=" + a + ", b=" + b + ", c=" + c + ", d=" + d

+ ", correctanswer=" + correctanswer + "]";

}

}

**Quiz.java**

package com.bean;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToMany;

@Entity

public class Quiz {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private int selfId;

private int quizid;

private String title;

//@OneToMany

//@JoinColumn(name = "qid")

private int qid;

public int getSelfid() {

return selfId;

}

public void setSelfId(int quizid) {

this.selfId = selfId;

}

public int getQuizid() {

return quizid;

}

public void setQuizid(int quizid) {

this.quizid = quizid;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public int getQid() {

return qid;

}

public void setQid(int qid) {

this.qid = qid;

}

@Override

public String toString() {

return "Quiz [quizid=" + quizid + ", title=" + title + ", qid=" + qid + "]";

}

}

**UserQuiz.java**

package com.bean;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "userquiz")

public class UserQuiz {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private int userid;

private String emailid;

private int quizid;

private int qid;

private String correctuseranswer;

public String getEmailid() {

return emailid;

}

public void setEmailid(String emailid) {

this.emailid = emailid;

}

public int getQuizid() {

return quizid;

}

public void setQuizid(int quizid) {

this.quizid = quizid;

}

public int getQid() {

return qid;

}

public void setQid(int qid) {

this.qid = qid;

}

public String getCorrectuseranswer() {

return correctuseranswer;

}

public void setCorrectuseranswer(String correctuseranswer) {

this.correctuseranswer = correctuseranswer;

}

public int getUserid() {

return userid;

}

@Override

public String toString() {

return "UserQuiz [userid=" + userid + ", emailid=" + emailid + ", quizid=" + quizid + ", qid=" + qid

+ ", correctuseranswer=" + correctuseranswer + "]";

}

public UserQuiz() {

super();

// TODO Auto-generated constructor stub

}

}

**Com.dao**

**QuizDao.java**

package com.dao;

import java.util.List;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.EntityTransaction;

import javax.persistence.Query;

import javax.transaction.Transaction;

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

import org.springframework.stereotype.Repository;

import com.bean.Participants;

import com.bean.Questions;

import com.bean.Quiz;

import com.bean.UserQuiz;

@Repository

public class QuizDao {

@Autowired

EntityManagerFactory emf;

public int partipantSignUp(Participants pt) {

try {

EntityManager manager = emf.createEntityManager();

EntityTransaction tran = manager.getTransaction();

tran.begin();

manager.persist(pt);

tran.commit();

return 1;

} catch (Exception e) {

System.out.println(e);

return 0;

}

}

public List<Participants> partipantLoginIn() {

EntityManager manag = emf.createEntityManager();

Query qry = manag.createQuery("select pt from Participants pt"); // JPQL

List<Participants> listOfParticipants = qry.getResultList();

return listOfParticipants;

}

public int addQuestion(Questions q) {

try {

EntityManager manager = emf.createEntityManager();

EntityTransaction tran = manager.getTransaction();

tran.begin();

manager.persist(q);

tran.commit();

return 1;

} catch (Exception e) {

System.out.println(e);

return 0;

}

}

public Questions findQuestions(int qid){

EntityManager manager = emf.createEntityManager(); // Session in Hibernate

Questions q = manager.find(Questions.class, qid); // session.get(Employee.class,id)

return q;

}

public int createQuiz(Quiz qu) {

try {

EntityManager manager = emf.createEntityManager();

EntityTransaction tran = manager.getTransaction();

tran.begin();

manager.persist(qu);

tran.commit();

return 1;

} catch (Exception e) {

System.out.println(e);

return 0;

}

}

public int takeQuiz(UserQuiz uq) {

try {

EntityManager manager = emf.createEntityManager();

EntityTransaction tran = manager.getTransaction();

tran.begin();

manager.persist(uq);

tran.commit();

return 1;

} catch (Exception e) {

System.out.println(e);

return 0;

}

}

public String checkResult() {

EntityManager manag = emf.createEntityManager();

Query qry = manag.createQuery("select count(uq.userid) from Questions q,UserQuiz uq where q.correctanswer = uq.correctuseranswer");

List result1 = qry.getResultList();

Query qry1 = manag.createQuery("select uq.emailid from Questions q,UserQuiz uq where q.correctanswer = uq.correctuseranswer");

List result2 = qry1.getResultList();

result2.addAll(result1);

String s = "Emailid : " + result2.get(0) + "Score : " + result2.get(1);

return s;

}

}

**com.service**

**QuizService.java**

package com.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.bean.Participants;

import com.bean.Questions;

import com.bean.Quiz;

import com.bean.UserQuiz;

import com.dao.QuizDao;

@Service

public class QuizService {

@Autowired

QuizDao quizDao;

public String storeParticipant(Participants pt) {

if(quizDao.partipantSignUp(pt) > 0) {

return "Participant data Stored Successfully";

}

else {

return "Data didn't store";

}

}

public List<Participants> getAllParticipants() {

return quizDao.partipantLoginIn();

}

public String storeQuestion(Questions q) {

if(quizDao.addQuestion(q) > 0) {

return "Question added Successfully";

}

else {

return "Question didn't add";

}

}

public Questions findQuestions(int qid) {

return quizDao.findQuestions(qid);

}

public String createQuiz(Quiz qu) {

if(quizDao.createQuiz(qu) > 0) {

return "Quiz created Successfully";

}

else {

return "Quiz didn't created";

}

}

public String takeQuiz(UserQuiz uq) {

if(quizDao.takeQuiz(uq) > 0) {

return "UserQuiz created Successfully";

}

else {

return "UserQuiz didn't created";

}

}

public String checkResult() {

return quizDao.checkResult();

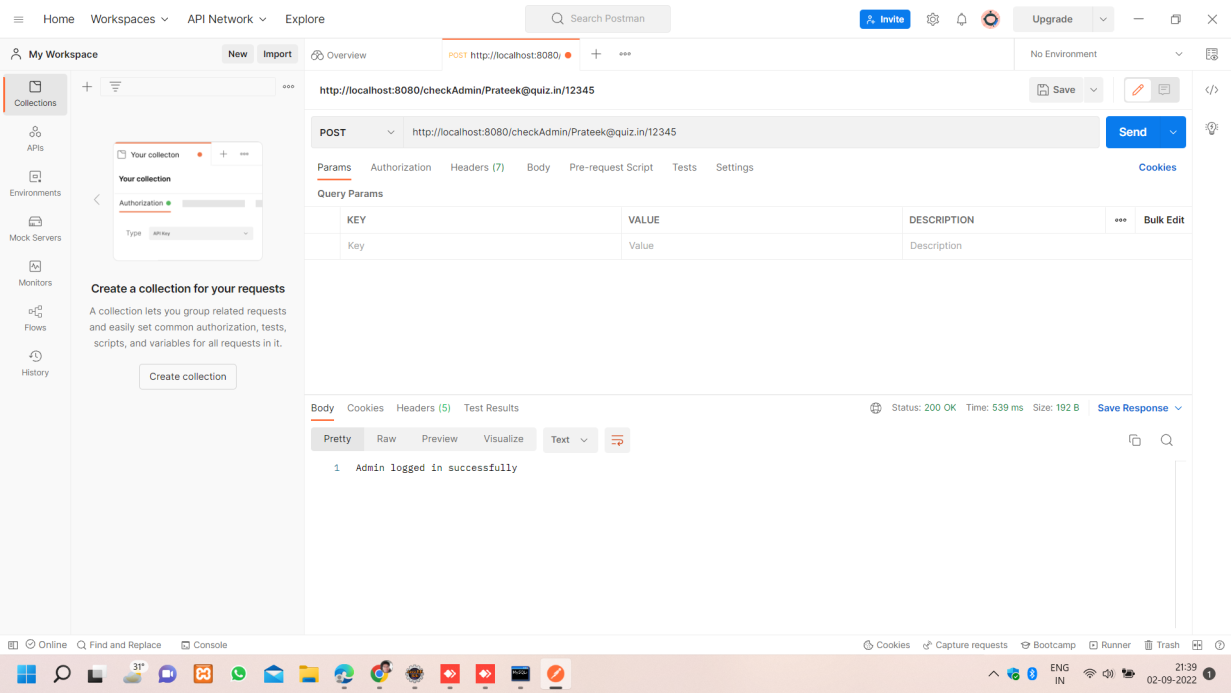
}

}

**Output**

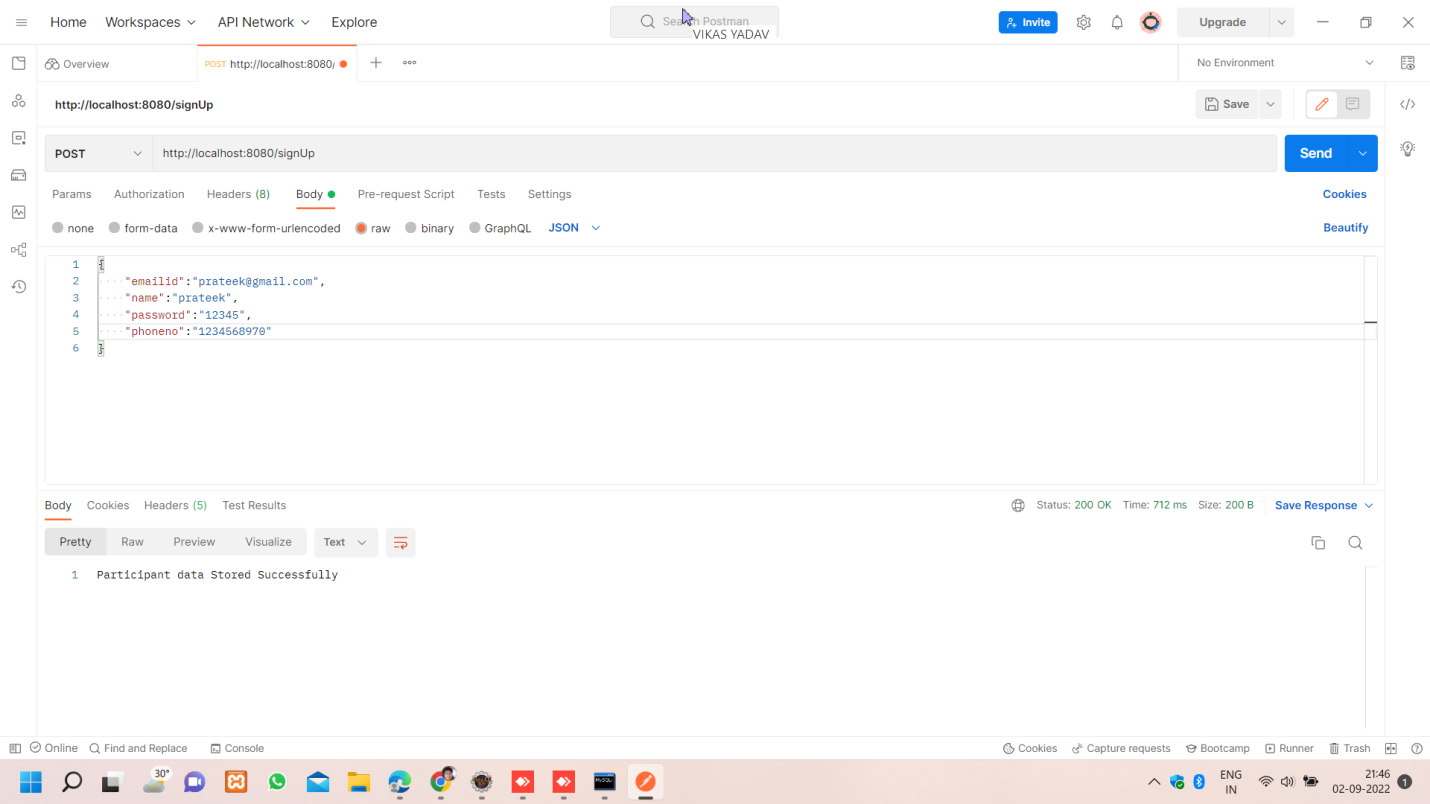
<http://localhost:8080/checkAdmin/>

This Url for checkAdmin in Postman tool



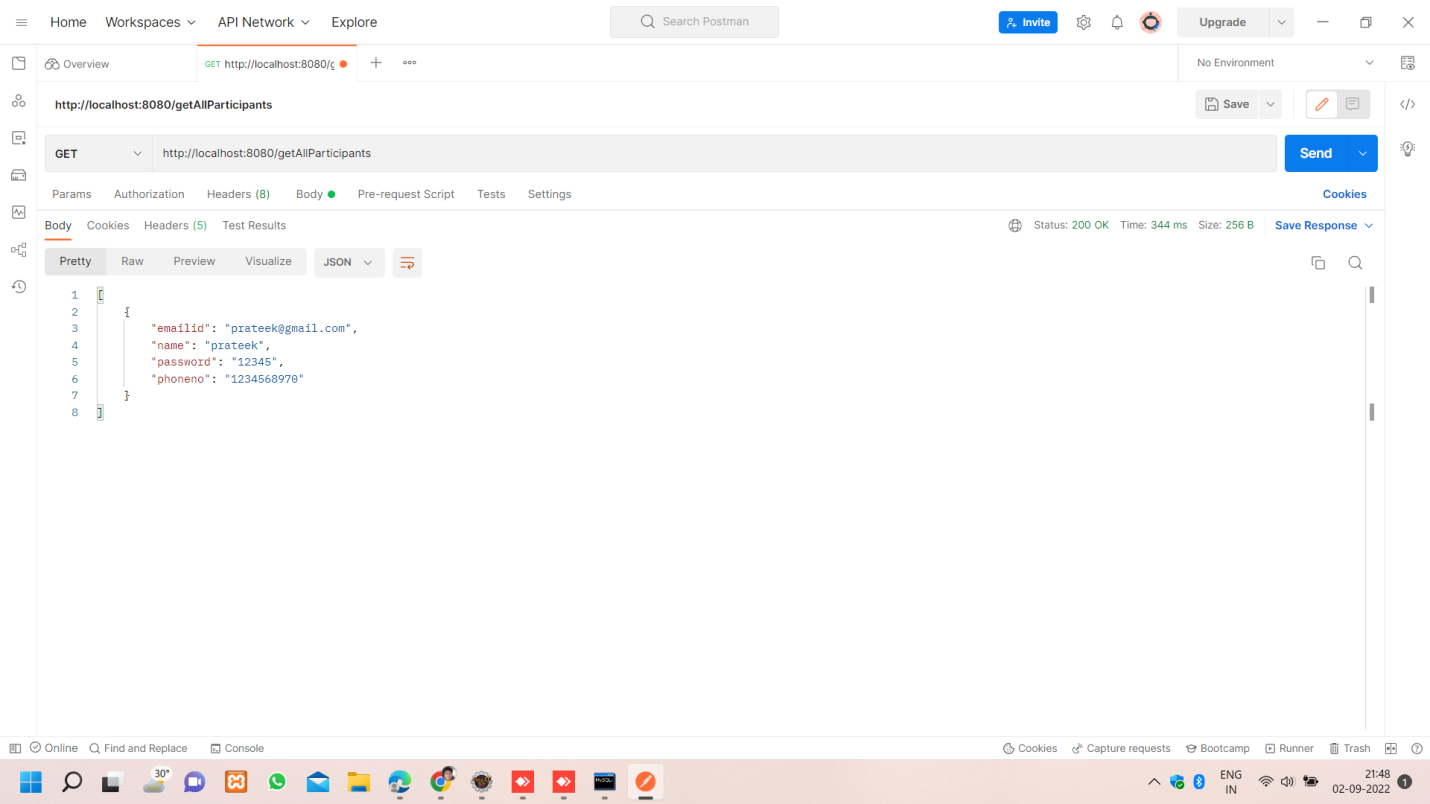
<http://localhost:8080/signUp/>

This URL shows that the participants signup in Online quiz portal.



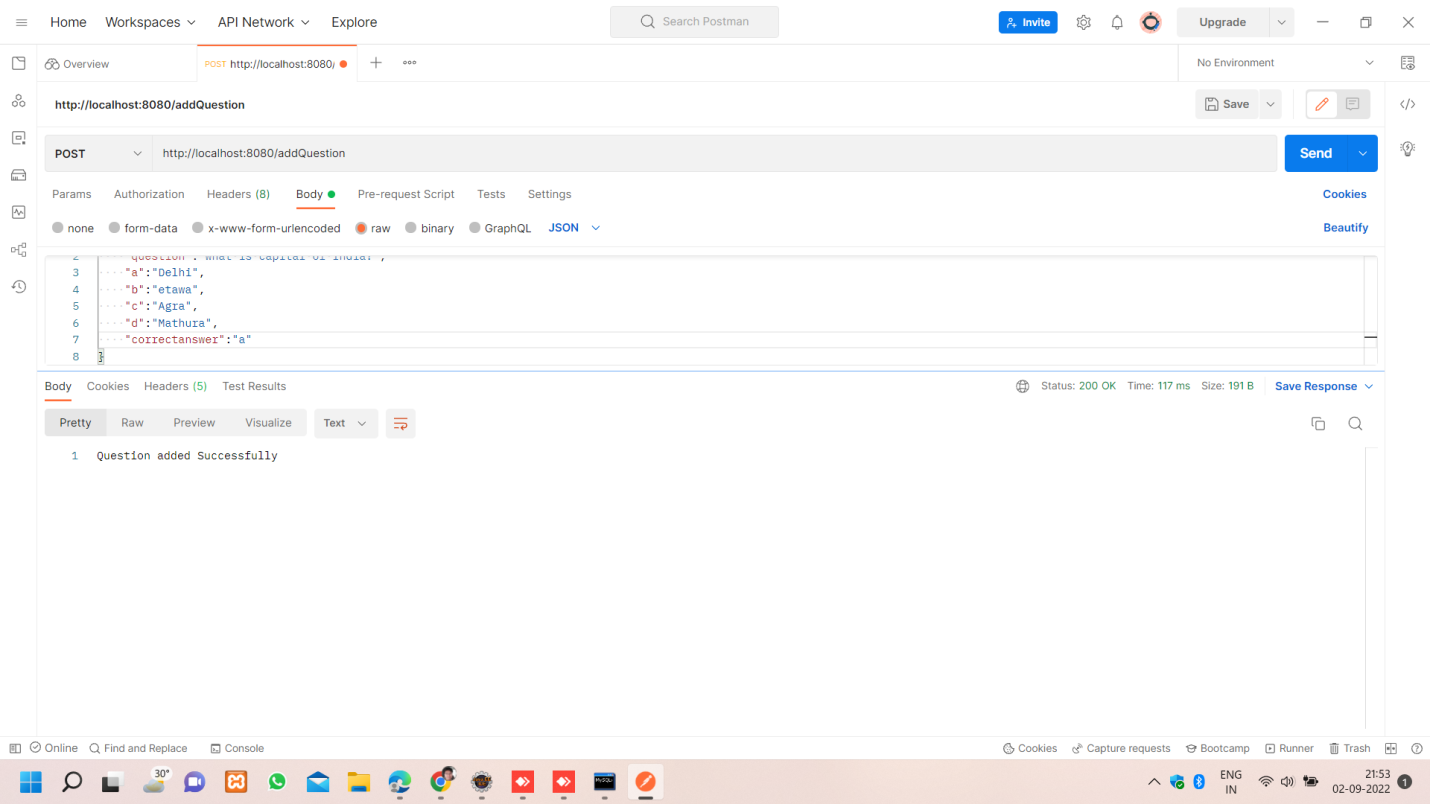
<http://localhost:8080/getAllParticipants>

This URL shows all the participants signup in Online quiz portal

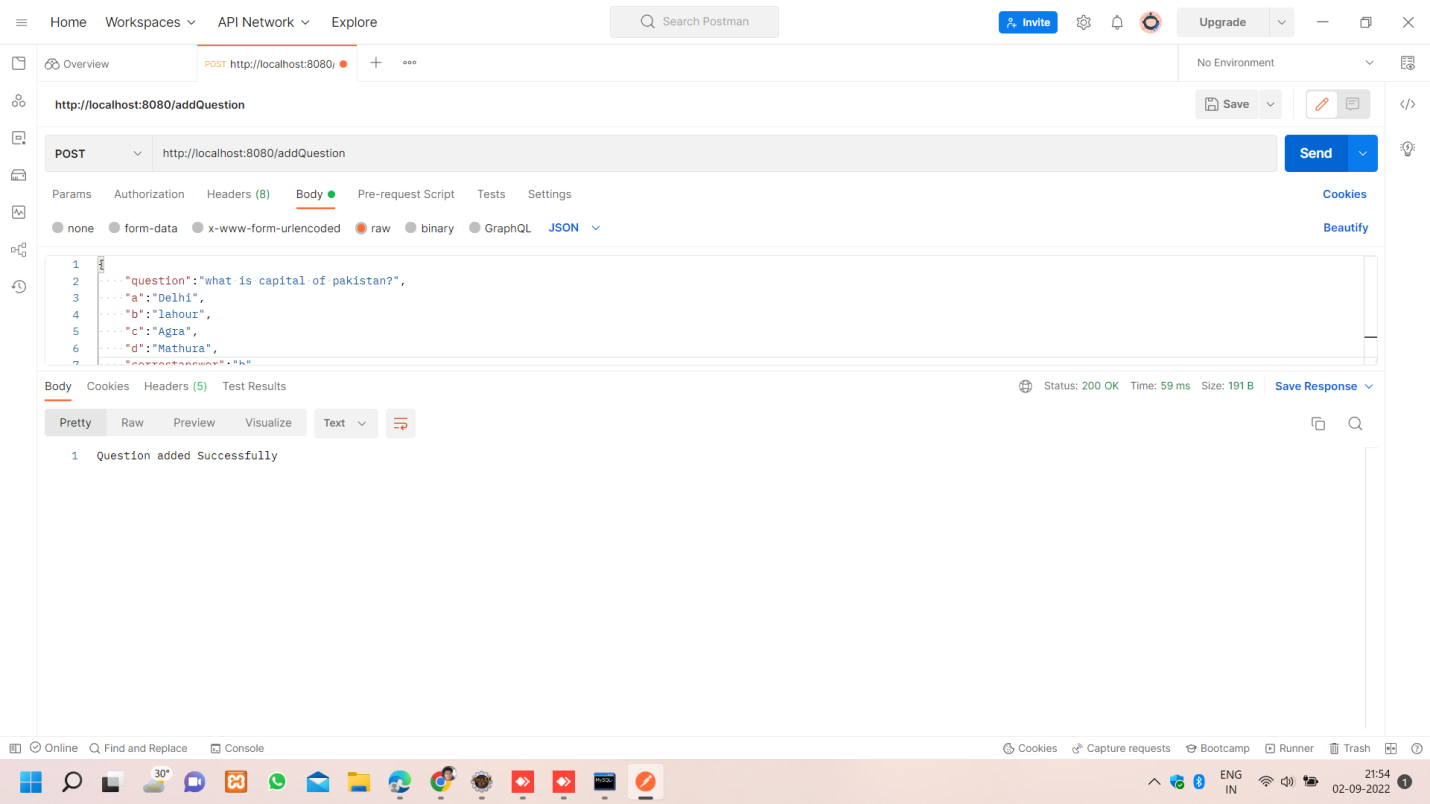


<http://localhost:8080/addQuestion/>

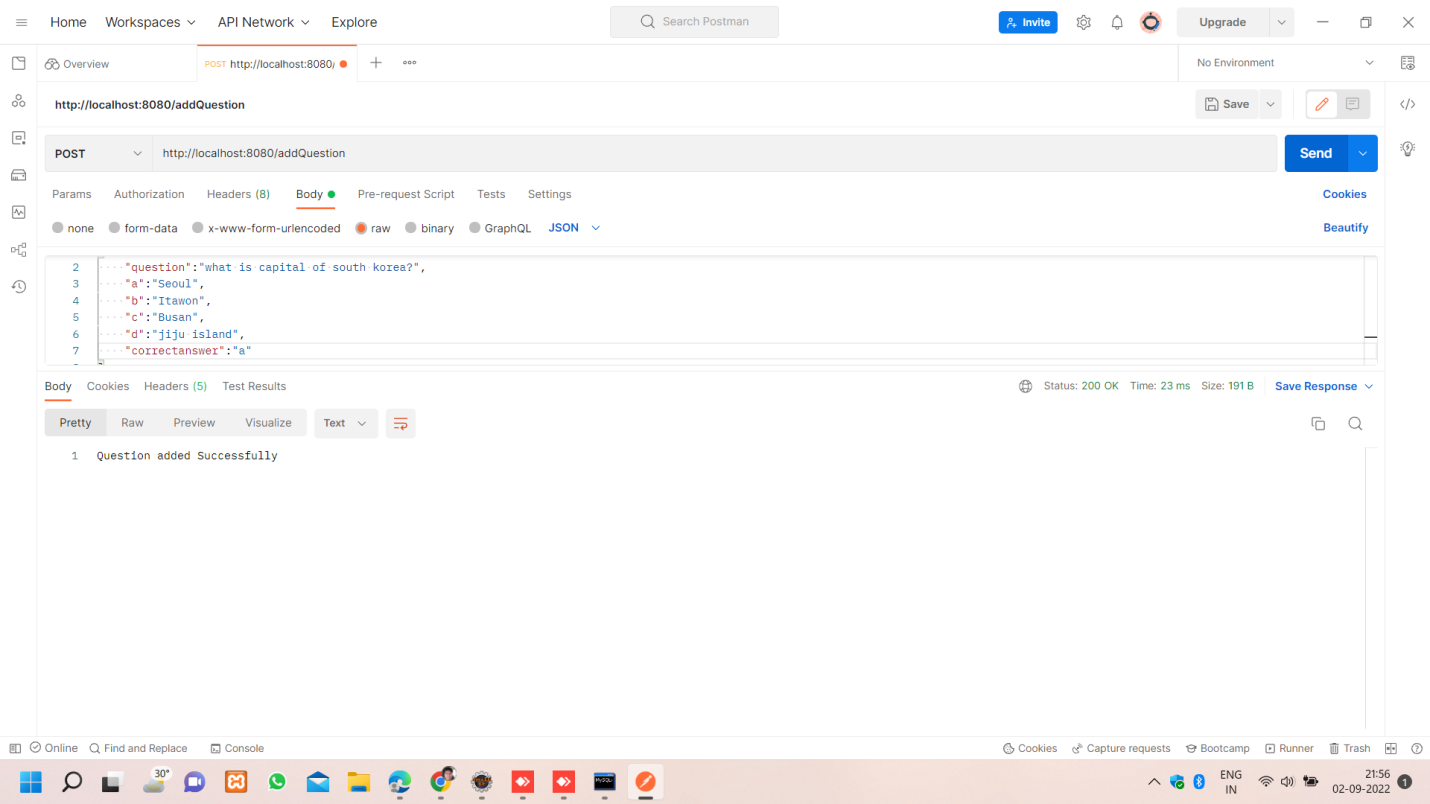
This URL shows that the add question in Online quiz portal.



First question inserted.



Second question inserted

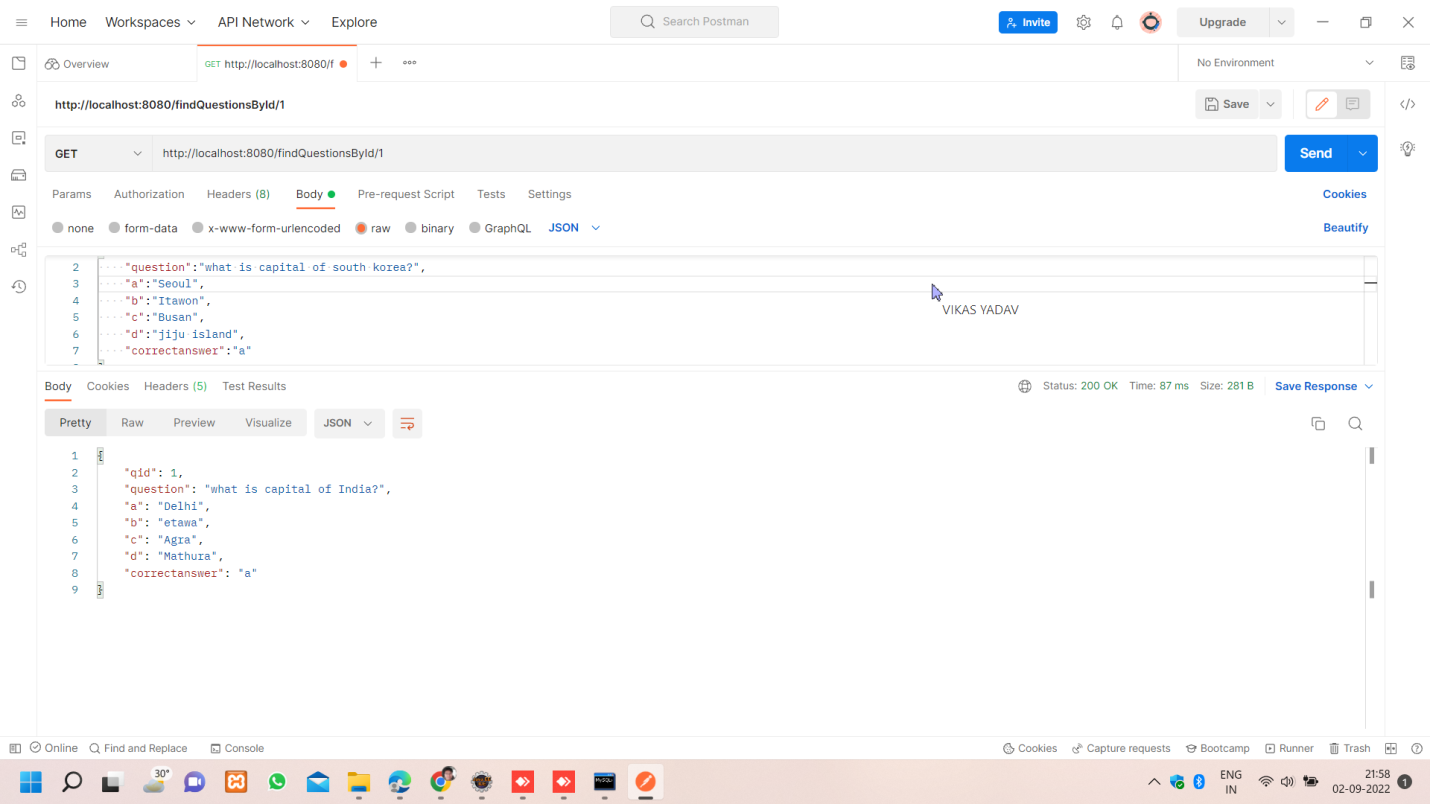


Third question inserted

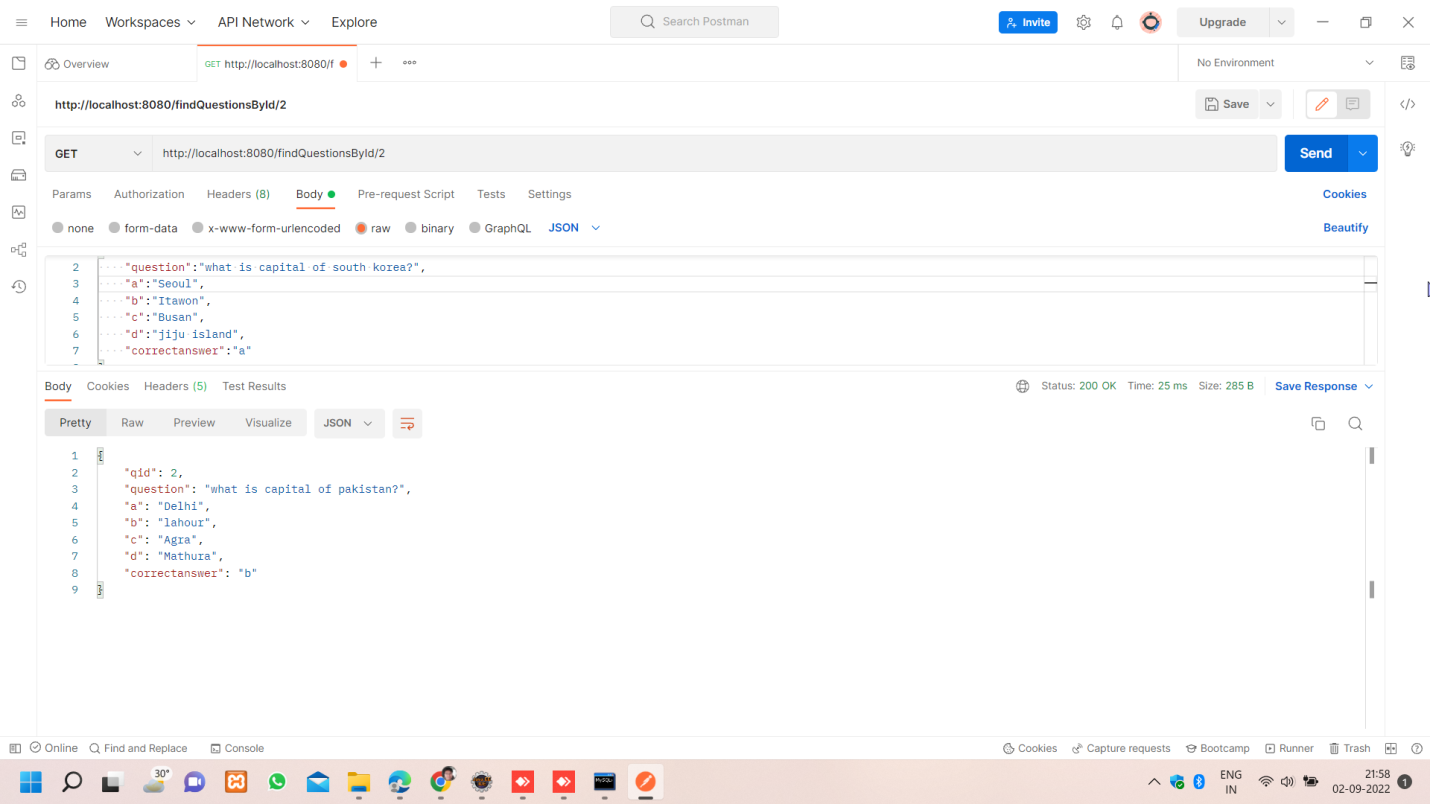
<http://localhost:8080/findQuestionsById/1>

This URL find question using by Id

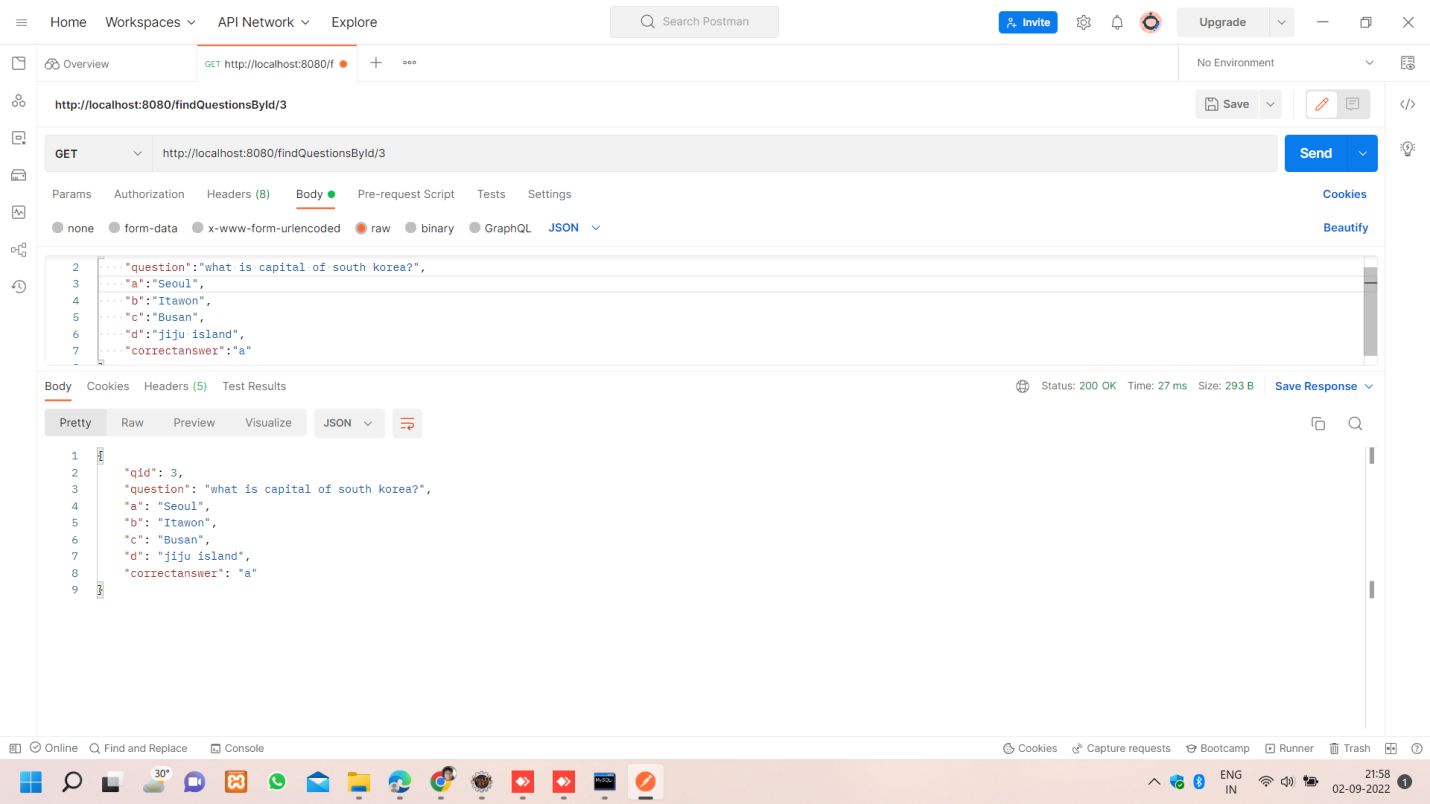
If you enter this Id=1 then you will get this output



If you enter this Id=2 then you will get this output



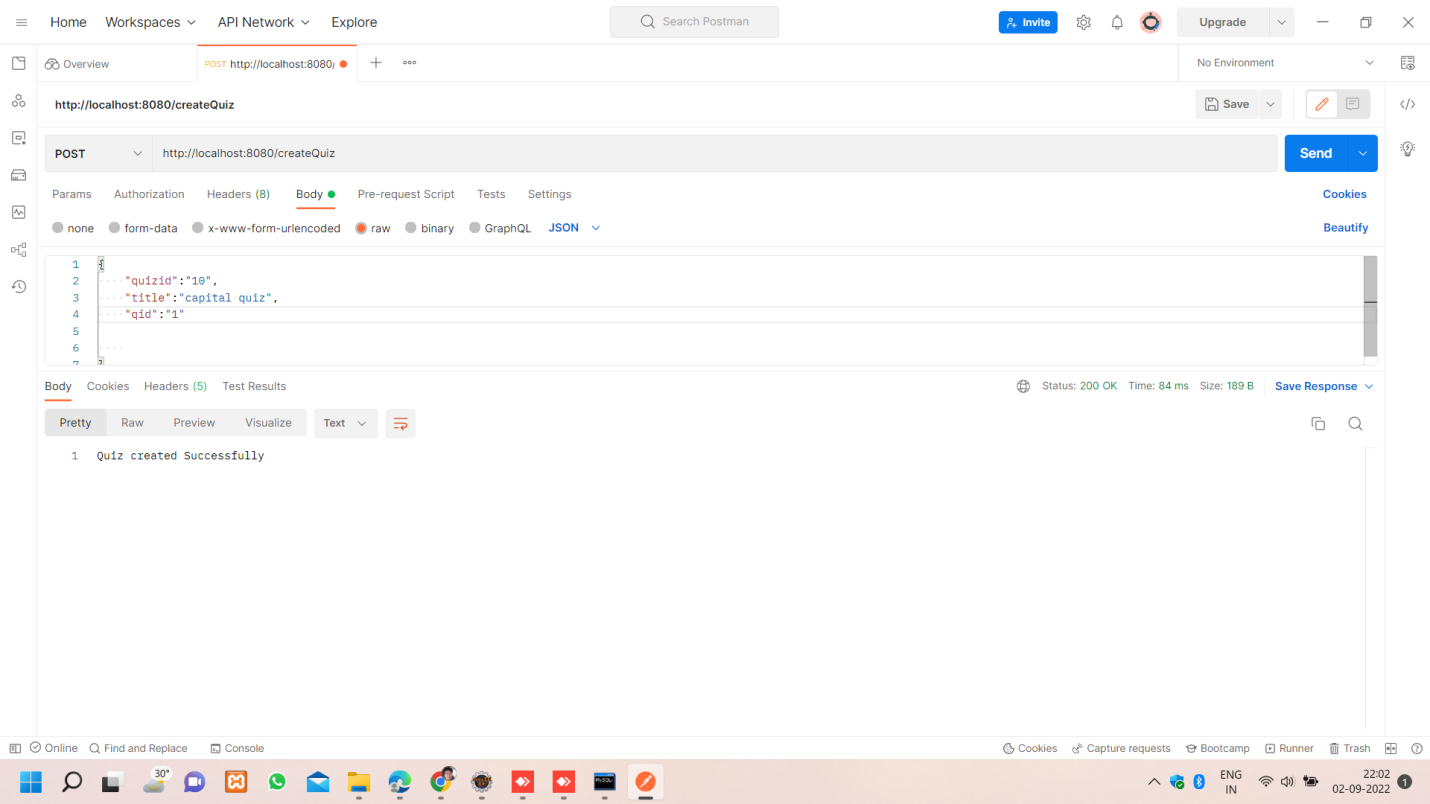
If you enter this Id=3 then you will get this output



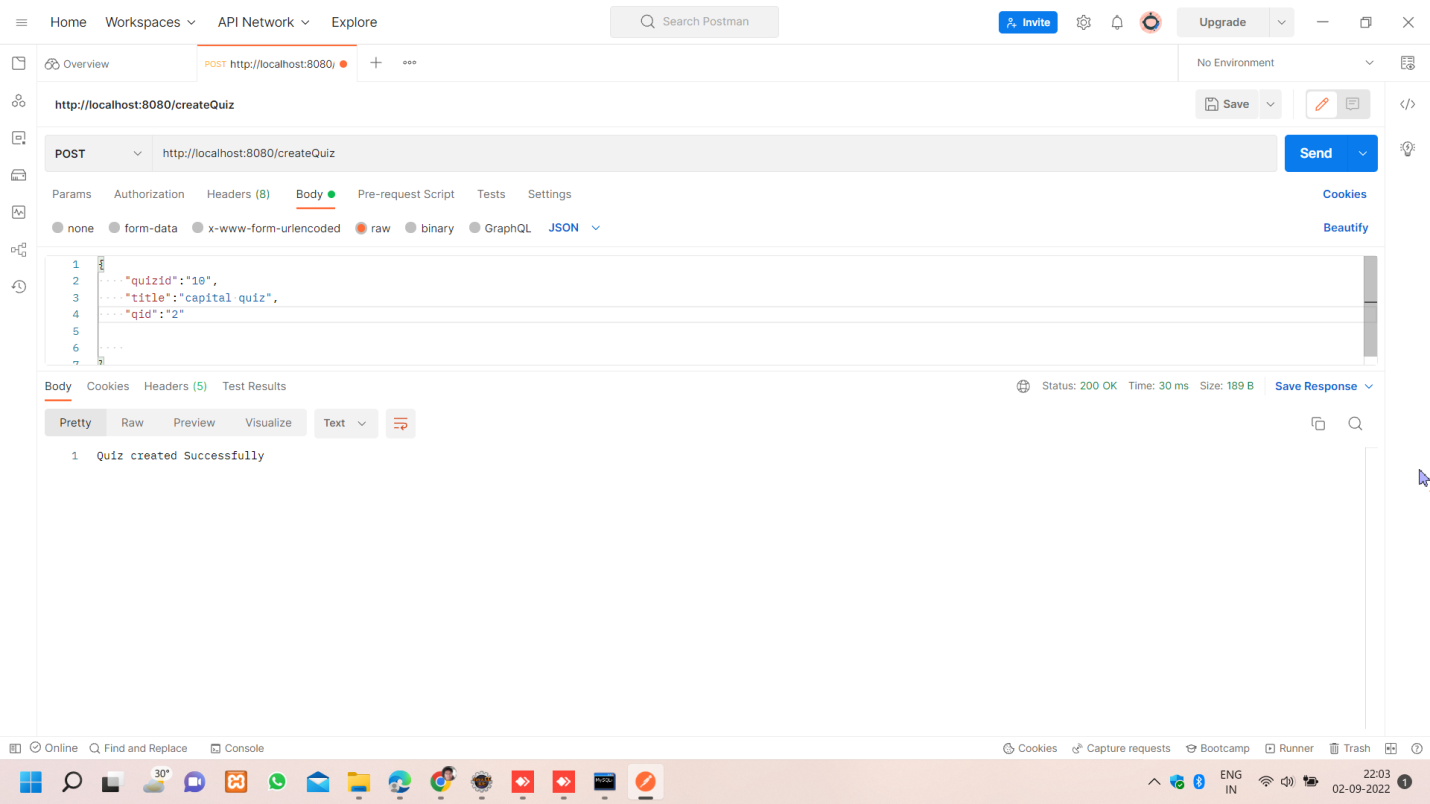
<http://localhost:8080/createQuiz/>

This url create quiz by using quiz id . Quiz id is unique and in this quiz we added questions by using question id

If you enter this qId=1 then you will get this output



If you enter this qId=2 then you will get this output

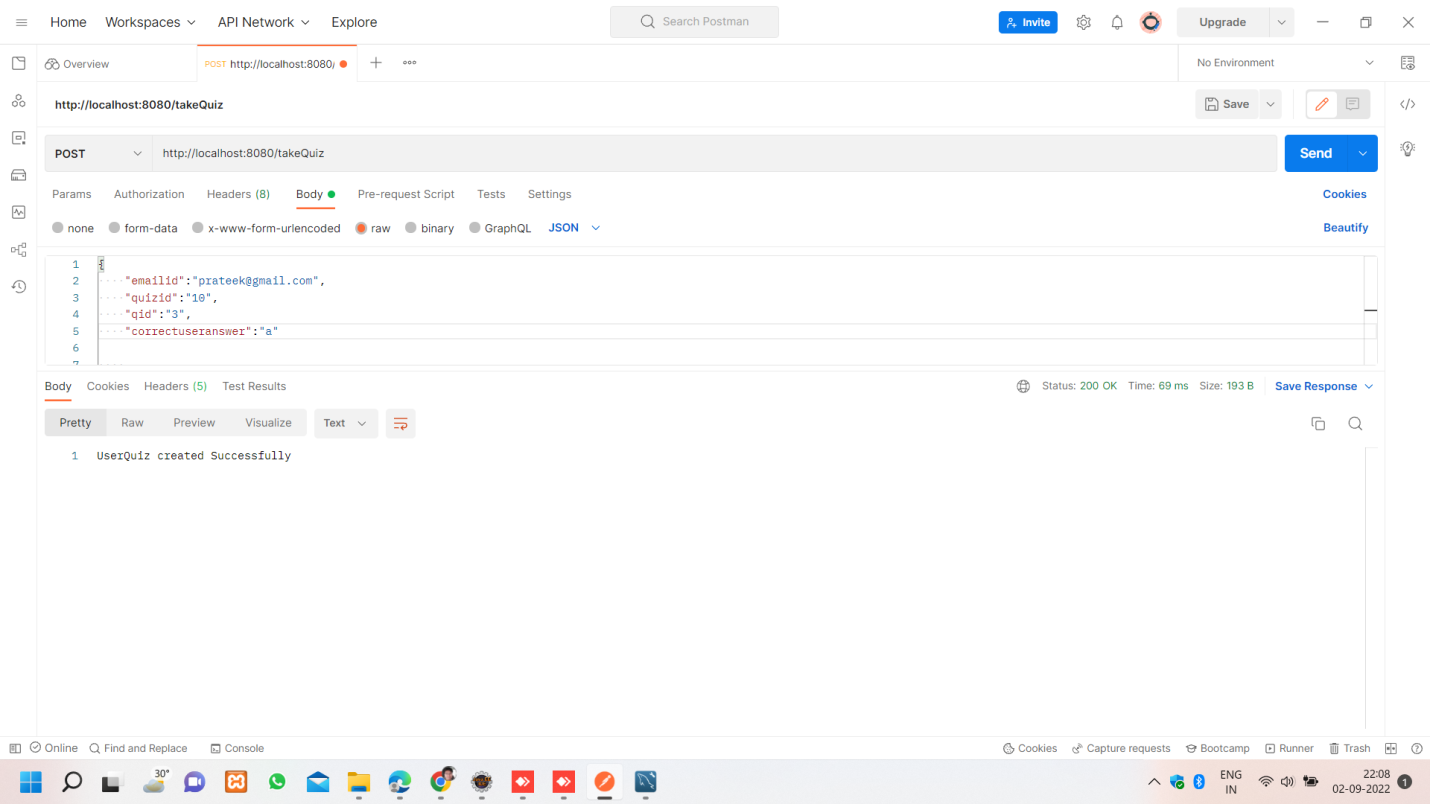


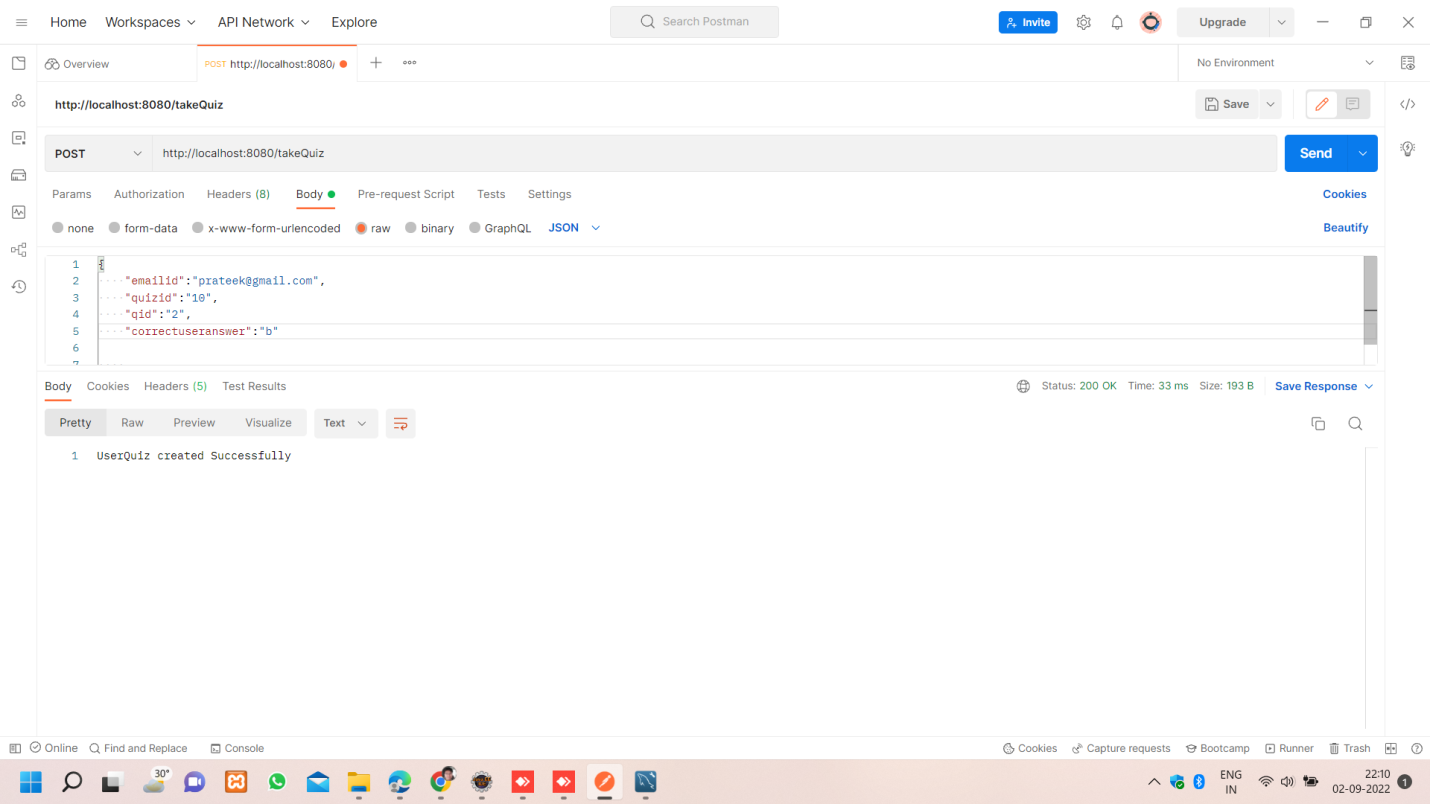
If you enter this qId=3 then you will get this output

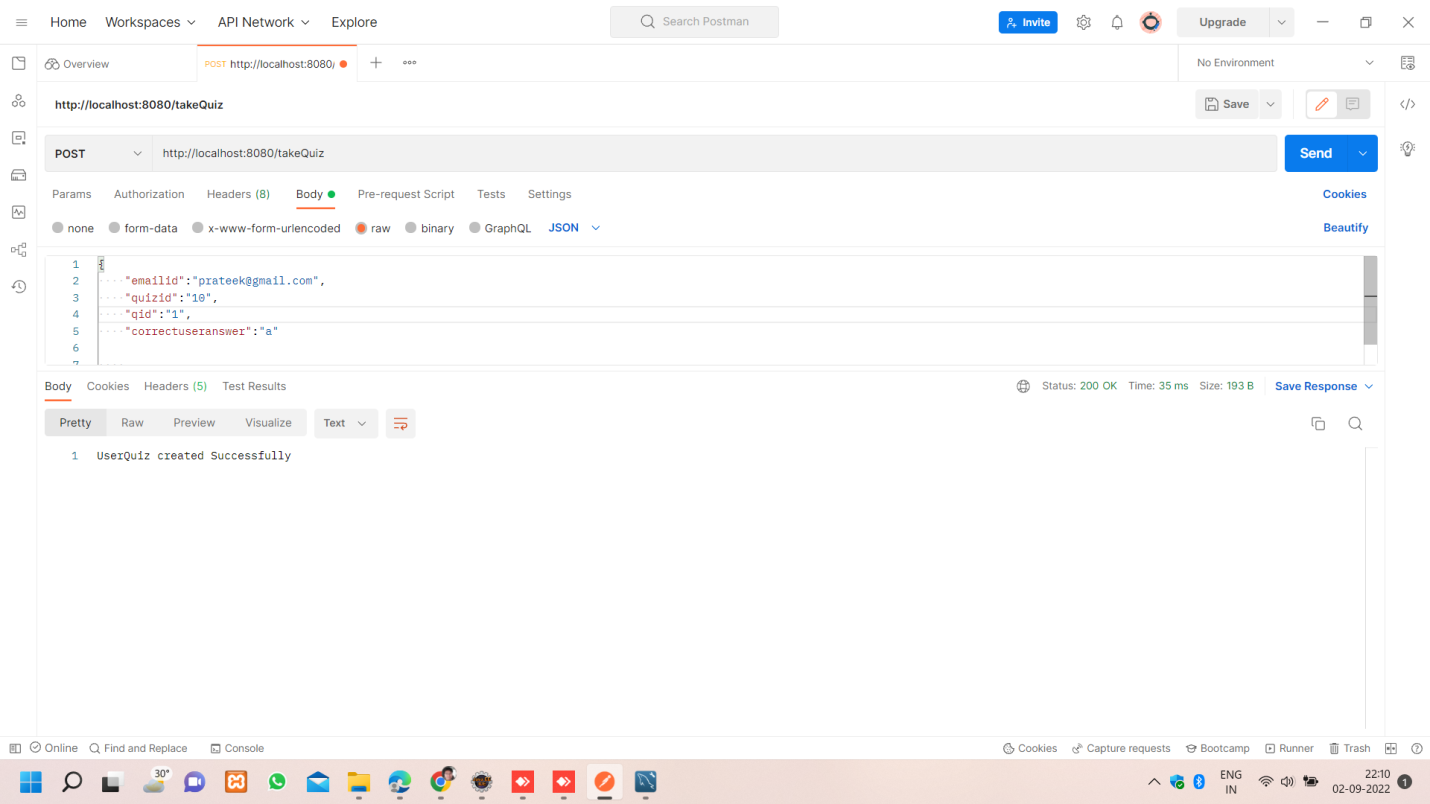


<http://localhost:8080/takeQuiz/>

when the participant gives the quiz , he/she will enter their email id , question id , quiz id and correct answer of the question inside the quiz.

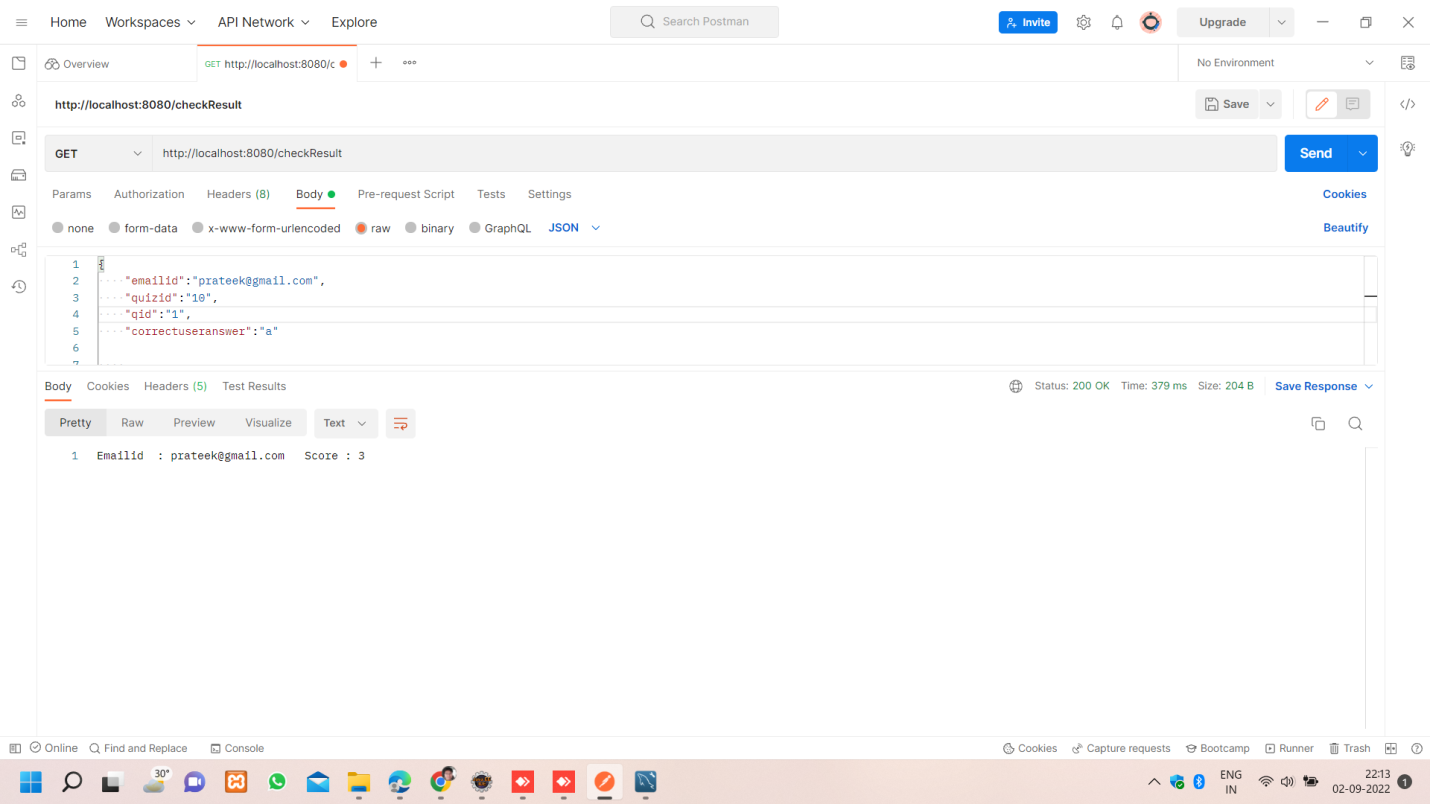




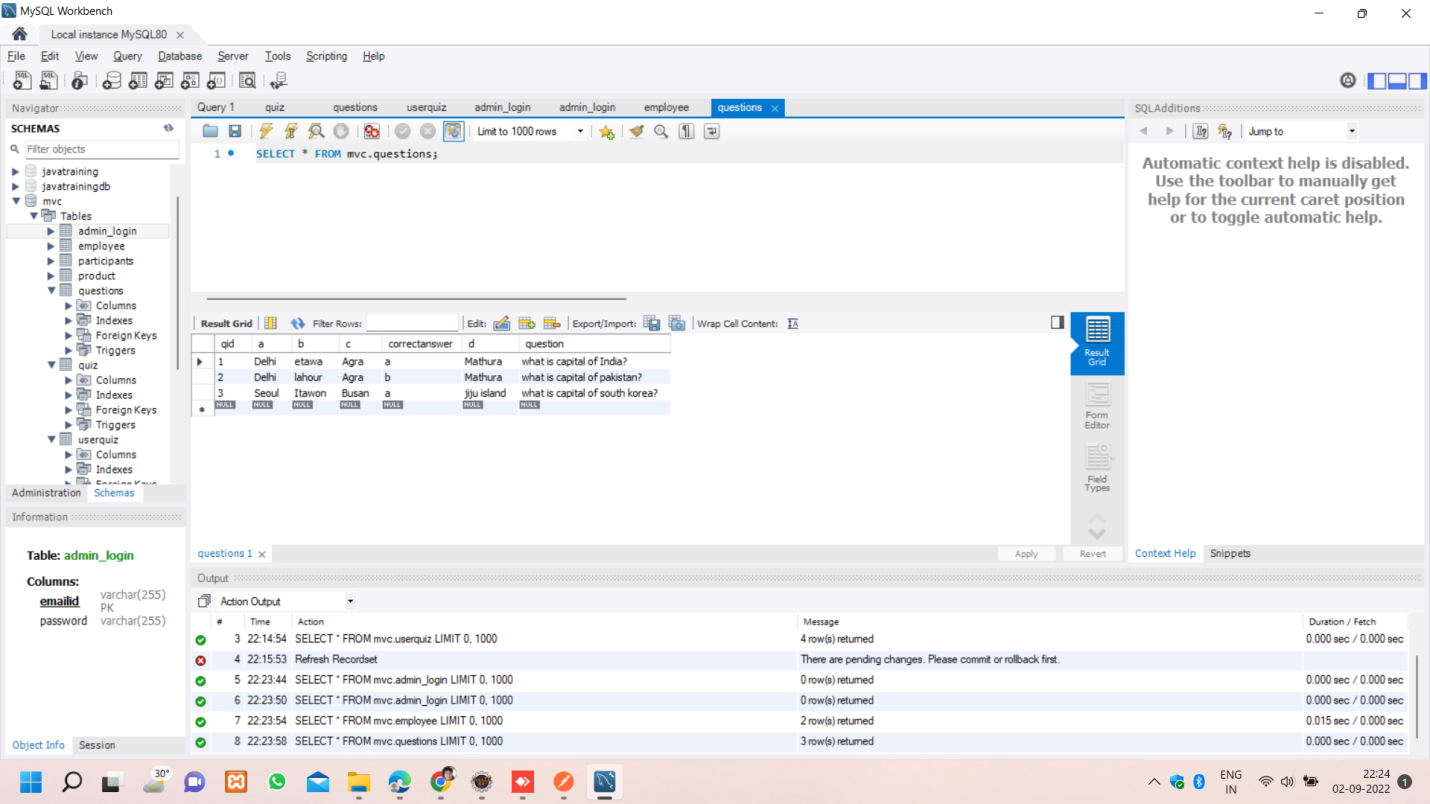


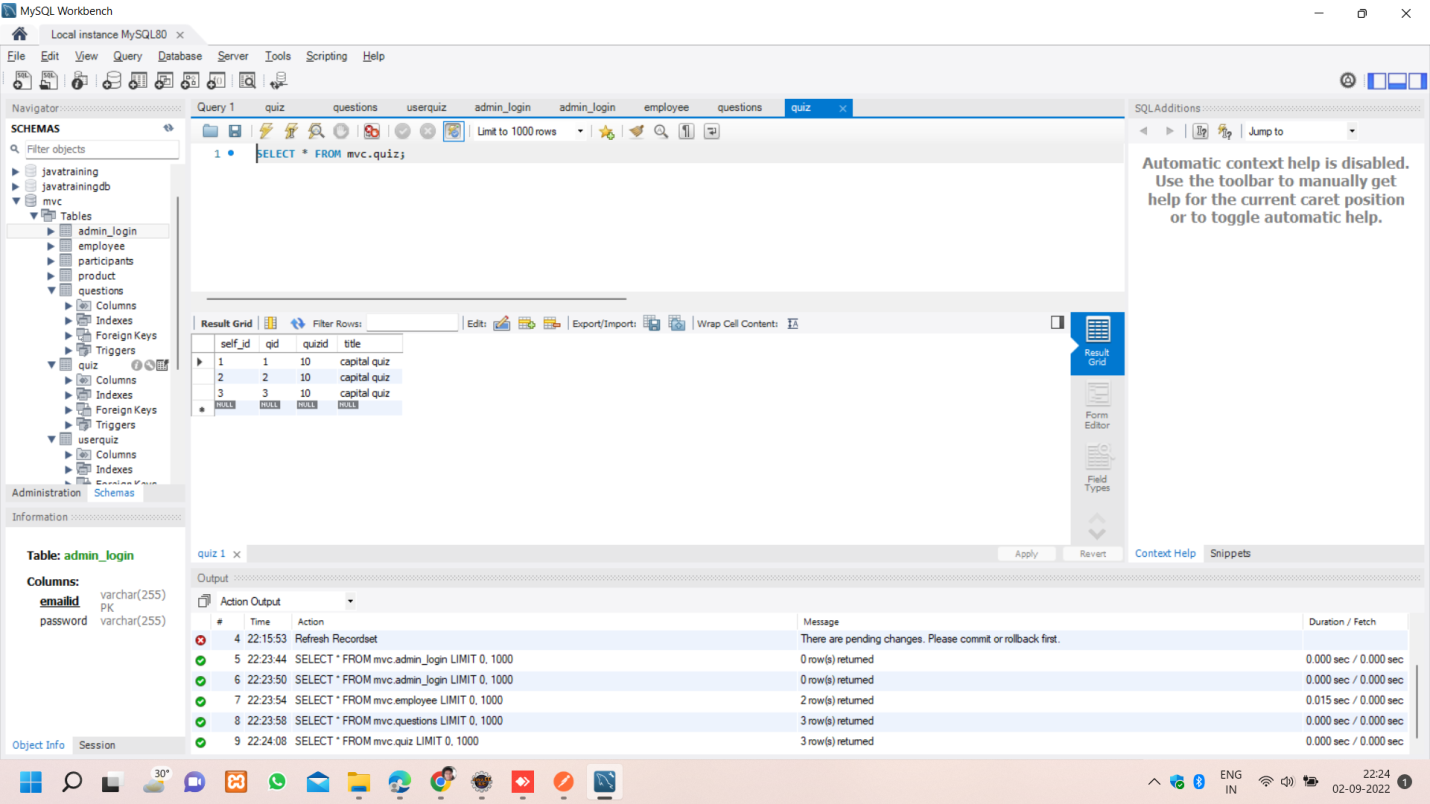
<http://localhost:8080/checkResult>

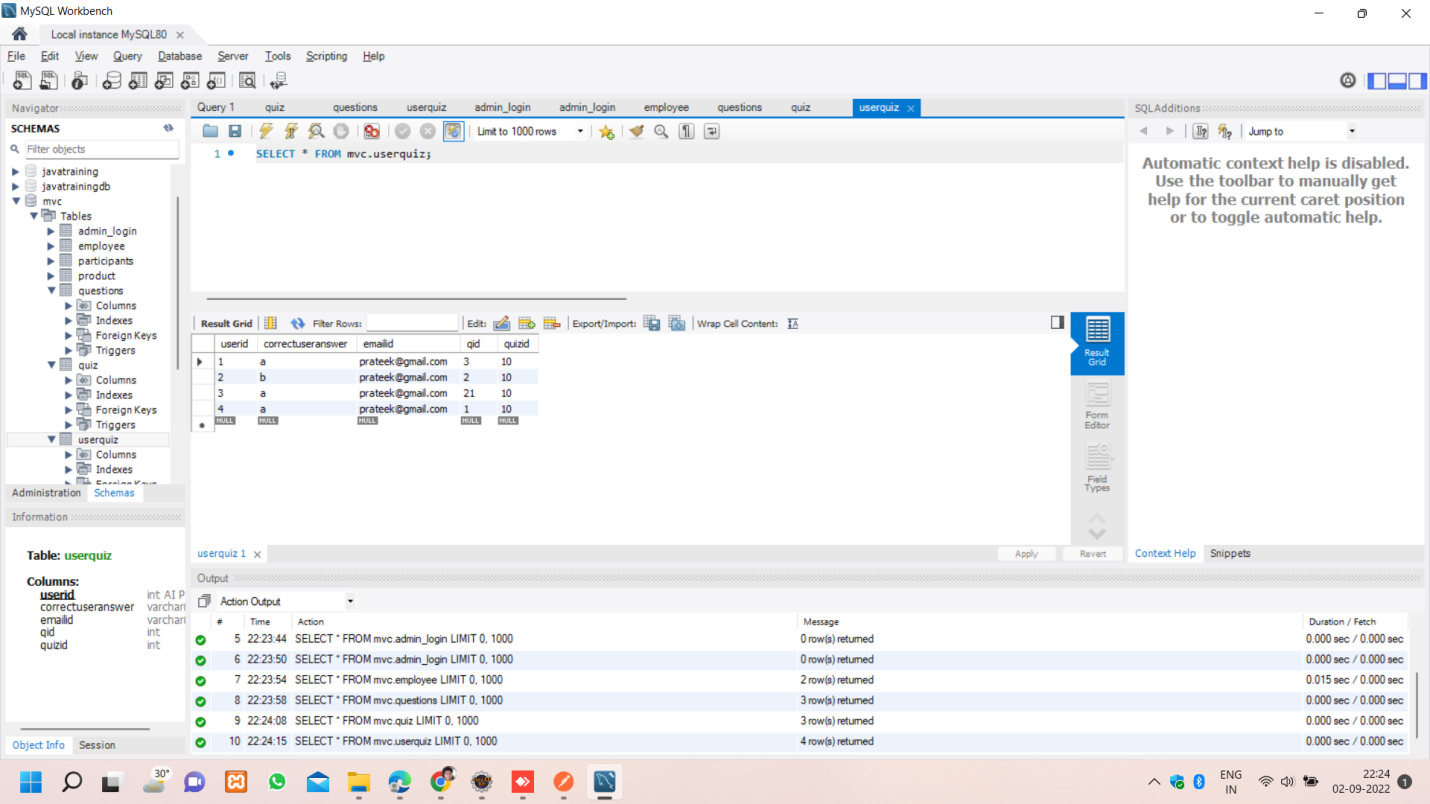
This URL shows the result of the quiz.



**Database**

****

****

****

**Application.properties**

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/mvc

spring.datasource.username=root

spring.datasource.password=Prateek#1974

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=update

**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>2.6.11</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>OnlineQuiz</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>OnlineQuiz</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>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</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>

**Conclusion**

* In this quiz admin creates a set of questions along with their answers.
* Once admin is authenticated, an access token is generated that can be used to add and modify quizzes, questions, and users.
* For creating a new quiz, the admin user enters a quizid and selects questions from the database using the questioned.
* Once the quiz is released, website users can start taking it.
* The user uses the register API to create an account The user attempts the quiz using quizid and gives the possible answers.
* After completing the quiz, the user checks the scores and compares their standings with other users.