==========

ONLINE QUIZ PORTAL

==========

JUNIT TESTING :

```
package com.quiz;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.junit.jupiter.api.MethodOrderer.OrderAnnotation;
import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.TestMethodOrder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.mock.web.MockMultipartFile;
import org.springframework.web.multipart.MultipartFile;
import static org.testng.Assert.assertEquals;
import com.quiz.bean.Login;
import com.quiz.bean.Questions;
import com.quiz.bean.Report;
import com.quiz.bean.Schedule;
import com.quiz.bean.Topics;
import com.quiz.bean.Users;
import com.quiz.service.QuizService;
import com.quiz.service.ReportService;
import com.quiz.service.ScheduleService;
import com.quiz.service.TopicService;
import com.quiz.service.UserService;
```

```
@TestMethodOrder(OrderAnnotation.class)
@SpringBootTest
class OnlineQuizPortalApplicationTests {

	@Autowired
	UserService userService;

	@Autowired
	TopicService topicService;

	@Autowired
	QuizService quizService;

	@Autowired
	ReportService reportService;

	@Autowired
	ScheduleService scheduleService;

	boolean actualResult;
	boolean expectedResult;

	List<Users> userList1 = new ArrayList<>();
	List<Users> userList2 = new ArrayList<>();

	List<Topics> topicList1 = new ArrayList<>();
	List<Topics> topicList2 = new ArrayList<>();

	List<Questions> quizList1 = new ArrayList<>();
	List<Questions> quizList2 = new ArrayList<>();

	List<Report> reportList1 = new ArrayList<>();
	List<Report> reportList2 = new ArrayList<>();
```

```
        List<Schedule> testList1 = new ArrayList<>();
        List<Schedule> testList2 = new ArrayList<>();

        Users user1 = new Users(111, "Domnic", "Dev", "dd@gmail.com",
"7010436645", "Dom", "User", "11964");
        Users user2 = new Users(112, "Mahesh", "kumar",
"mahesh@gmail.com", "9876987654", "MK", "User", "19181");

        Users admin1 = new Users(1, "Sara", "Ameer", "sara@gmail.com",
"7010436645", "sk", "Admin", "300");
        Users admin2 = new Users(2, "Virat", "Ameer", "vk@gmail.com",
"8978675645", "virat", "Admin", "18");

        Login response1 = new Login("User Login Successfully");
        Login response2 = new Login("Admin Login Successfully");
        Login response3 = new Login("Login Fail");

        Topics topic1 = new Topics(4, "ANGULAR");
        Topics topic2 = new Topics(4, "JS");

        Questions quiz2 = new Questions(5, 1, "Who invented Java
Programming?", "Guido van Rossum", " James Gosling",
                        "Dennis Ritchie", "Bjarne Stroustrup", " James Gosling");
        Questions quiz3 = new Questions(4, 1, "Who invented Java
Programming?", "Guido van Rossum", " James Gosling",
                        "Dennis Ritchie", "Bjarne Stroustrup", " James Gosling");

        Report report1 = new Report(2, 110, "Java", 80, "Great Grade!!!");
        Report report2 = new Report(1, 110, "HTML", 100, "Great Grade!!!");
```

```
        Schedule test1 = new Schedule(4, "Java", "11:00:00", "2024-02-12",
"Easy");
        Schedule test2 = new Schedule(4, "HTML", "11:30:00", "2024-02-13",
"Hard");

        @Test
        @Order(1)
        void contextLoads() {
        }

        @Test
        @Order(2)
        void performUserLogin() {
                Login login1 = userService.login(user1);
                response1.equals(login1);
        }

        @Test
        @Order(3)
        void performUserLoginFail() {
                Login login2 = userService.login(user2);
                response3.equals(login2);
        }

        @Test
        @Order(4)
        void performAdminLogin() {
                Login login3 = userService.login(admin1);
                response2.equals(login3);
        }

        @Test
        @Order(5)
        void performAdminLoginFail() {
                Login login4 = userService.login(admin2);
```

```
		response3.equals(login4);
	}

	@Test
	@Order(6)
	void performUserRegister() {
		Users user3 = new Users(121, "KL", "Rahul", "klr@gmail.com",
"9089786756", "KLR", "User", "11968");
		expectedResult = true;
		actualResult = userService.registerUser(user3);
		assertEquals(expectedResult, actualResult);
	}

	@Test
	@Order(7)
	void performUpdateUser() {
		Users user4 = new Users(120, "KL", "Rahul", "klr@gmail.com",
"9089786756", "KLRahul", "User", "11968");
		expectedResult = true;
		actualResult = userService.updateUser(user4);
		assertEquals(expectedResult, actualResult);
	}

	@Test
	@Order(8)
	void performDeleteUser() {
		expectedResult = true;
		actualResult = userService.deleteUser((long) 120);
		assertEquals(expectedResult, actualResult);
	}

	@Test
	@Order(9)
	void performGetAllUsers() {
		userList1 = userService.getAllUsers();
```

```
        userList2.add(user1);
        userList2.add(admin1);
        userList2.equals(userList1);
}

@Test
@Order(10)
void performGetAllCandidates() {
        userList1 = userService.getAllCandidates();
        userList2.add(user1);
        userList2.add(admin1);
        userList2.equals(userList1);
}

@Test
@Order(11)
void performAddTopic() {
        expectedResult = true;
        actualResult = topicService.addTopic(topic1);
        assertEquals(expectedResult, actualResult);
}

@Test
@Order(12)
void performUpdateTopic() {
        expectedResult = true;
        actualResult = topicService.update(topic2);
        assertEquals(expectedResult, actualResult);
}

@Test
@Order(13)
void performDeleteTopic() {
        expectedResult = true;
        actualResult = topicService.deleteTopics((long) 4);
```

```
            assertEquals(expectedResult, actualResult);
        }

        @Test
        @Order(14)
        void performGetAllTopics() {
            topicList1 = topicService.getAllTopics();
            topicList2.add(topic1);
            topicList2.add(topic2);
            topicList2.equals(topicList1);
        }

        @Test
        @Order(15)
        void performAddQuiz() throws IOException {
            FileInputStream inputFile = new
FileInputStream("D:\\JavaOnlineQuiz (2).xlsx");
            MockMultipartFile file = new MockMultipartFile("file",
"test1.txt", "multipart/form-data", inputFile);
            expectedResult = true;
            actualResult = quizService.addQuiz(file);
            assertEquals(expectedResult, actualResult);
        }

        @Test
        @Order(16)
        void performUpdateQuiz() {
            expectedResult = true;
            actualResult = quizService.updateQuiz(quiz2);
            assertEquals(expectedResult, actualResult);
        }
```

```
@Test
@Order(17)
void performDeleteQuiz() {
	expectedResult = true;
	actualResult = quizService.deleteQuiz((long) 6);
	assertEquals(expectedResult, actualResult);
}

@Test
@Order(18)
void performGetAllQuizzes() {
	quizList1 = quizService.getAllQuizzes();
	quizList2.add(quiz2);
	quizList2.add(quiz3);
	quizList2.equals(quizList1);
}

@Test
@Order(19)
void performAddReport() {
	expectedResult = true;
	actualResult = reportService.addReport(report1);
	assertEquals(expectedResult, actualResult);
}

@Test
@Order(20)
void performUpdateReport() {
	expectedResult = true;
	actualResult = reportService.updateReport(report2);
	assertEquals(expectedResult, actualResult);
}
```

```
@Test
@Order(21)
void performDeleteReport() {
	expectedResult = true;
	actualResult = reportService.deleteReport((long) 4);
	assertEquals(expectedResult, actualResult);
}

@Test
@Order(22)
void performGetAllReports() {
	reportList1 = reportService.getAllReport();
	reportList2.add(report1);
	reportList2.add(report2);
	reportList2.equals(reportList1);
}

@Test
@Order(23)
void performAddTest() {
	expectedResult = true;
	actualResult = scheduleService.addTest(test1);
	assertEquals(expectedResult, actualResult);
}

@Test
@Order(24)
void performUpdateTest() {
	expectedResult = true;
	actualResult = scheduleService.updateSchedule(test2);
	assertEquals(expectedResult, actualResult);
}
```

```
	@Test
	@Order(25)
	void performDeleteTest() {
		expectedResult = true;
		actualResult = scheduleService.deleteTest((long) 4);
		assertEquals(expectedResult, actualResult);
	}

	@Test
	@Order(26)
	void performGetAllTests() {
		testList1 = scheduleService.getAllTests();
		testList2.add(test1);
		testList2.add(test2);
		testList2.equals(testList1);
	}
}
```

All test cases are passed successfully :