**WRITEUP**

**Step 1:** To create a spring project named as OnlineQuizPortal

**Step 2:** In src/main/java create a package named com.bean, com.controller,

com.demo, com.repository, com.service.

**Step 3:** In package com.bean create a class named admin, question, quiz, result,

statistics, test, user, and write the code respectively.

**Step 4:** In package com.controller create a class named MainController and write

the code.

**Step 5:** In package com.demo create a class name OnlineQuizPortalRestApplication and

write the code.

**Step 6**: In package com.repository create a class named AdminRepo,

QuestionRepo, Quizrepo, Resultrepo, Testrepo, Userrepo, and write the code.

**Step 7:** In package com.service create a class named AdminSer, UserSer, and write

the code.

**Step 8:** In src/main/java create a package named com.demo.

**Step 9:** In package com.demo a class name OnlineQuizPortalRestApplication write the

code.

**Step 10:** In MYSQL, create a database Quiz and show the tables.

**Step 11:** Save the files and run the project in the Run as Java application.

**Step 12:** Enter the URL (http://localhost:8080/) with the respective file name in a

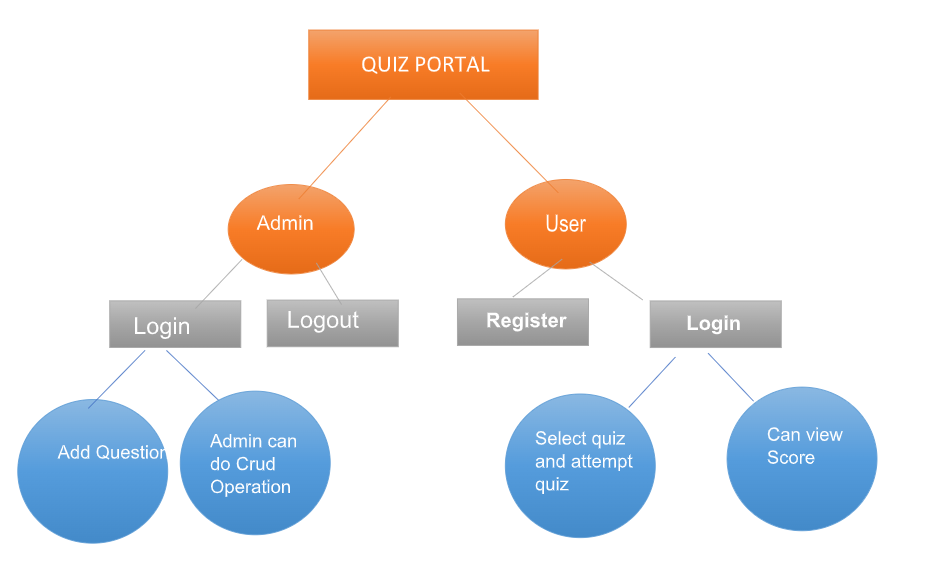
postman.

**STEP 13:** Click Clone repository and clone in the git repository of the Eclipse.

**STEP 14:** Stage the file from unstage and commit the message.

**STEP 15:** Push the OnlineQuizePortal file in GitHub through Eclipse.

**FlowChart**

****

* Includes a RESTful Web API to perform CRUD operations on Domain objects as per requirement using Spring Boot and MySQL/Oracle database.
* The admin user has a separate API to access the admin portal, which requires authentication with the admin username and password.
* The admin user can update the profile details and change the password after using the access token generated by login.
* The admin user can add questions using the **addQuestions** API.
* The admin user can create a quiz by entering **quizid** and selecting questions with **questionid**.
* The admin user can obtain statistics on total quizzes, questions, and users by using their APIs.
* The admin user can find the users who participated in the quiz along with scores and standings.
* Users can explore various quizzes created by the admin.
* Users can create their profile using new user registration.
* Users can take the quiz and try to answer the questions.
* Users can also check if the given answers are correct, as well as their results and positions.

**Admin User Scenario:**

* The admin wants to create an online quiz for the website users.
* The admin creates a set of questions along with their answers.
* Admin opens login API and logs in with the admin username and password.
* Once admin is authenticated, an access token is generated that can be used to add and modify quizzes, questions, and users.
* The admin creates a quiz by providing a name and an id.
* For creating a new quiz, the admin user enters a **quizid** and selects questions from the database using the **questionid**.
* After adding the questions, the admin runs the API and makes the quiz available to its users.
* Once the quiz is released, website users can start taking it.

**Participants** **Scenario:**

* The user uses the register API to create an account, which provides an access token.
* After registering an account, the user logs in with the access token to take any quiz.
* The user browses various quizzes created by Admin using the **getAllQuizzes** API.
* The user attempts the quiz using **quizid** and gives the possible answers.
* The user views whether the provided answers are right or wrong. The correct answer is highlighted differently so that they are easily identified.
* After completing the quiz, the user checks the scores and compares their standings with other users.