Online Quiz Portal

Algorithm:

```
1.Import the necessary packages and classes required for the servlets and database operations.
2. Create a servlet named Add SQL Question Servlet.
  Override the doPost() method to handle the form submission.
Retrieve the question, options, and correct option from the request parameters.
        --> Create a new SQL Question object and set its properties.
        --> Open a session using Hibernate Util and begin a transaction.
        --> Save the SQL Question object to the database using the session.
        --> Commit the transaction and close the session.
        --> Redirect back to the add question.jsp page.
3. Create a servlet named Delete SQL Question Servlet.
        --> Override the doGet() method to handle the delete operation.
        --> Retrieve the question ID from the request parameter.
        --> Delete the SQL question from the database using SQL Question DAO.
        --> Redirect back to the view page.
4. Create a servlet named Edit SQL Question Servlet.
        --> Override the doGet() method to retrieve the question for editing.
        --> Retrieve the question ID from the request parameter.
        --> Retrieve the SQL question from the database using SQL Question DAO.
        --> Forward the question object to the edit sqlquestion.jsp page for editing.
        --> Override the doPost() method to handle the update operation.
```

--> Retrieve the question ID from the request parameter.

--> Retrieve the SQL question from the database using SQL QuestionDAO.

--> Update the SQL question with the form data.

--> Update the SQL question in the database using SQLQuestionDAO.

--> Redirect to the SQL question list page.

5. Create a class named SQL Question.

--> Annotate the class with @Entity and define the table name.

--> Define the properties of the SQL question including ID, question, options, and correct

option.

--> Implement getter and setter methods for the properties.

--> Implement a method to format the options as a string.

6. Create a class named SQL Question DAO.

--> Implement a method to get all SQL questions from the database.

--> Implement a method to format the options of a SQL question as a string.

--> Implement a method to get a SQL question by ID.

--> Implement a method to delete a SQL question by ID.

--> Implement a method to update a SQL question.

7. Create a servlet named SQL Test Servlet.

--> Override the doGet() method to retrieve all SQL questions from the database.

--> Forward the SQL questions to the sql Test.jsp page for rendering.

8. Create a servlet named Submit Answer Servlet.

--> Override the doPost() method to handle the submission of user answers.

--> Create a map to store the user answers.

--> Get the submitted answers from the request parameters.

--> Calculate the score by comparing the user answers with the correct options.

--> Set the score as a request attribute and forward to the sql Test Result.jsp page.

9. Create a servlet named View SQL Question Servlet.

--> Override the doGet() method to retrieve all SQL questions from the database.

--> Forward the SQL questions to the view sql list.jsp page for rendering.

10. User Registration:

--> The user fills out the registration form with their name, email, password, and date of birth.

--> The `Register Servlet` receives the form data via a POST request.

--> The servlet creates a new `User` object with the provided data.

--> The `UserDAO` class is used to save the user in the database using Hibernate.

--> The user is redirected to the "userdashboard.jsp" page upon successful registration.

11. User Login:

--> The user enters their email and password in the login form.

-->The `UserLoginServlet` receives the login credentials via a POST request.

--> The servlet uses the `UserDAO` class to validate the user's credentials against the stored data in the database.

--> If the credentials are valid, a session is created, and the user is redirected to the "userdashboard.jsp" page.

--> If the credentials are invalid, an error message is displayed, and the user is redirected

back to the login page.

12. Admin Login:

--> Similar to user login, the admin enters their email and password in the login form.

--> The `AdminLoginServlet` receives the admin credentials via a POST request.

--> The servlet uses the `AdminDAO` class to validate the admin's credentials against the stored data in the database.

--> If the credentials are valid, a session attribute "admin Authenticated" is set to true, and the admin is redirected to the "admindashboardpage.jsp" page.

--> If the credentials are invalid, an error message is displayed, and the admin is redirected back to the admin login page.

13. User Management (View, Edit, Delete):

--> The "viewuser.jsp" page displays a list of users.

--> The admin can click on the "Edit" button next to a user to edit their details.

--> The `EditUserServlet` receives a GET request with the user ID as a parameter.

--> The servlet uses the `UserDAO` class to fetch the user details based on the ID and forwards the user object to the "edituser.jsp" page for editing.

-->The admin can make changes to the user details and submit the form.

--> The `UpdateUserServlet` receives a POST request with the updated user data.

-->The servlet uses the `UserDAO` class to update the user in the database.

--> The admin is redirected back to the "viewuser.jsp" page to see the updated user list.

--> The admin can click on the "Delete" button next to a user to delete them.

--> The `DeleteUserServlet` receives a GET request with the user ID as a parameter.

--> The servlet uses the `UserDAO` class to delete the user from the database.

-->The admin is redirected back to the "viewuser.jsp" page to see the updated user list.

14. Logout:

--> The admin or user can click on the "Logout" button to log out of the system.

--> The `LogoutServlet` receives a GET request.

--> The servlet invalidates the current session and redirects the user to the "loggedout.jsp" page.

15. `Edit user.jsp`: This file displays a form to edit user details like name and email. The form is submitted to the "update user" action.
16. `Addquestion.jsp`: It shows a table of subjects and provides links to add and view questions for each subject. Currently, it only displays options for the "SQL" subject.
17. `Addquestionsql.jsp`: This file displays a form to add SQL questions. It includes input fields for the question, options, and correct option.
18. `Admindashboardpage.jsp`: It is the admin dashboard page that provides links to add questions and view the user list.
19. `Adminlogin.jsp`: This file displays an admin login form with email and password fields.
20. `Editsqlquestion.jsp`: It shows a form to edit an SQL question. The existing question details are pre-filled in the form.
21. `Error.jsp`: This file displays an error message.
22. `Header.jsp`: It includes common header content that can be reused across multiple JSP pages.
23. `Index.jsp`: This is the main page of the application, which provides links for user registration, user login, and admin login.
24. `Loggedout.jsp`: This page is displayed after a user logs out, showing a message that the session has terminated.
25. `Sqltest.jsp`: It displays SQL test questions fetched from the database. The user can select answers for each question and submit the form.
26. `Sqltestresult.jsp`: This page shows the result of the SQL test, displaying the user's score.

27. `Userdashboard.jsp`: It is the user dashboard page, displaying a welcome message and providing a link to take the SQL test.

28. `Userlogin.jsp`: This file contains a user login form with email and password fields.

29.userquizdetails.jsp: This JSP file displays a table of user details, including their ID, name, email, and score. It retrieves the user details from the request attribute.

30.userregister.jsp: This JSP file displays a form for user registration. It takes the user's name, email, password, and date of birth as input.

31.viewsqllist.jsp: This JSP file displays a table of SQL questions. It retrieves a list of SQL questions and dynamically generates the table rows for each question.

32.Viewuser.jsp: Retrieve a list of users from the database. Display the user ID, name, email, date of birth, and action links for editing and deleting.