PROJECT ID – 65HIBKJS

Project Title: Online Quiz Application

Project Description:

Create an Online Quiz Application in Java that allows users to take quizzes on various topics. The application should support multiple-choice questions, track user progress, and provide feedback on quiz performance.

Project Setup:

- Create a java project
- Add MySQL JDBC Driver to the project
- Create a MySQL database and tables for users,quizzes,questions and results.

DATABASE SCHEMA:

```
CREATE DATABASE QuizApp;

USE QuizApp;

CREATE TABLE users (id INT AUTO_INCREMENT PRIMARY KEY, username VARCHAR(50) NOT NULL UNIQUE, password VARCHAR(255) NOT NULL);

CREATE TABLE quizzes ( id INT AUTO_INCREMENT PRIMARY KEY, title VARCHAR(255) NOT NULL);

CREATE TABLE questions ( id INT AUTO_INCREMENT PRIMARY KEY, quiz_id INT, question TEXT,
    option1 VARCHAR(255),
    option2 VARCHAR(255),
    option3 VARCHAR(255),
    option4 VARCHAR(255),
    correct_option INT, FOREIGN KEY (quiz_id) REFERENCES quizzes(id) );


CREATE TABLE results (id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    quiz_id INT,
    score INT,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (quiz_id) REFERENCES quizzes(id) );
```

Project Requirements:

1. User Authentication:

- Implement a login system where users can create an account or log in with existing credentials.

// Create a user interface for registration and login using JavaFx

User.java

```
public class User {
    private int id;
    private String username;                    //create a user class with attributes ‘username’,’password’
    private String password;
public class User {
    private int id;
    private String username;
    private String password;                                // Getters and Setters
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {                           // use hashing to store password
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
}
```

UserDAO.java

```
import java.sql.*;
public class UserDAO {
```

```
    private Connection conn;
public UserDAO(Connection conn) {                          //setup database table of users
        this.conn = conn;
    }
       public boolean registerUser(User user) throws SQLException {
        String sql = "INSERT INTO users (username, password) VALUES (?, ?)";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, user.getUsername());
        stmt.setString(2, user.getPassword());                          // Use hashed password
        int rowsInserted = stmt.executeUpdate();
        return rowsInserted > 0;
    }
       public User loginUser(String username, String password) throws SQLException {
        String sql = "SELECT * FROM users WHERE username = ? AND password = ?";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, username);
        stmt.setString(2, password);                          // Use hashed password for comparison
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            User user = new User();
            user.setId(rs.getInt("id"));
            user.setUsername(rs.getString("username"));
            return user;
        }
            return null;
    }
}
```

2. Quiz Management:

- Allow administrators to create quizzes with multiple-choice questions.
- Each question should have a title, options, and correct answer(s).
- Enable administrators to edit and delete quizzes.

//provide an interface for administrators to create,edit,and delete quizzes.

Quiz.java

```
public class Quiz {                         //Create quiz class
    private int id;
    private String title;
    public int getId() {                    //Setters and getters
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
}
```

Question.java

```
public class Question {                     //Create question class
    private int id;
    private int quizId;
    private String question;
    private String option1;
    private String option2;
    private String option3;
    private String option4;
    private int correctOption;

    public int getId() {                            //Setters and Getters
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getQuizId() {
        return quizId;
    }
    public void setQuizId(int quizId) {
        this.quizId = quizId;
    }
    public String getQuestion() {
        return question;
    }
    public void setQuestion(String question) {
        this.question = question;
    }

    public String getOption1() {
        return option1;
    }
    public void setOption1(String option1) {
        this.option1 = option1;
    }
    public String getOption2() {
        return option2;
    }
    public void setOption2(String option2) {
        this.option2 = option2;
```

```
    }

    public String getOption3() {
        return option3;
    }
    public void setOption3(String option3) {
        this.option3 = option3;
    }

    public String getOption4() {
        return option4;
    }
    public void setOption4(String option4) {
        this.option4 = option4;
    }

    public int getCorrectOption() {
        return correctOption;
    }
    public void setCorrectOption(int correctOption) {
        this.correctOption = correctOption;
    }
}
```

QuizDAO.java:

```
import java.sql.*;

import java.util.ArrayList;

import java.util.List;


public class QuizDAO {                    // implement CRUD operations for quizzes and questions

    private Connection conn;                        using JDBC

public QuizDAO(Connection conn) {

        this.conn = conn;

    }

      public boolean createQuiz(Quiz quiz) throws SQLException {

        String sql = "INSERT INTO quizzes (title) VALUES (?)";

        PreparedStatement stmt = conn.prepareStatement(sql);

        stmt.setString(1, quiz.getTitle());

        int rowsInserted = stmt.executeUpdate();

        return rowsInserted > 0;

    }


    public List<Quiz> getAllQuizzes() throws SQLException {

        String sql = "SELECT * FROM quizzes";
```

```
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);

        List<Quiz> quizzes = new ArrayList<>();
        while (rs.next()) {
            Quiz quiz = new Quiz();
            quiz.setId(rs.getInt("id"));
            quiz.setTitle(rs.getString("title"));
            quizzes.add(quiz);
        }
        return quizzes;
    }

    public boolean createQuestion(Question question) throws SQLException {
        String sql = "INSERT INTO questions (quiz_id, question, option1, option2, option3, option4,
correct_option) VALUES (?, ?, ?, ?, ?, ?, ?)";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setInt(1, question.getQuizId());
        stmt.setString(2, question.getQuestion());
        stmt.setString(3, question.getOption1());
        stmt.setString(4, question.getOption2());
        stmt.setString(5, question.getOption3());
        stmt.setString(6, question.getOption4());
        stmt.setInt(7, question.getCorrectOption());
        int rowsInserted = stmt.executeUpdate();
        return rowsInserted > 0;
    }
      public List<Question> getQuestionsByQuizId(int quizId) throws SQLException {
        String sql = "SELECT * FROM questions WHERE quiz_id = ?";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setInt(1, quizId);
        ResultSet rs = stmt.executeQuery();

        List<Question> questions = new ArrayList<>();
        while (rs.next()) {
```

```
            Question question = new Question();
            question.setId(rs.getInt("id"));
            question.setQuizId(rs.getInt("quiz_id"));
            question.setQuestion(rs.getString("question"));
            question.setOption1(rs.getString("option1"));
            question.setOption2(rs.getString("option2"));
            question.setOption3(rs.getString("option3"));
            question.setOption4(rs.getString("option4"));
            question.setCorrectOption(rs.getInt("correct_option"));
            questions.add(question);
        }

        return questions;
    }
}
```

3. Quiz Taking:

- Users should be able to select and take quizzes from the available list of topics.

- Display one question at a time with options for the user to select the answer(s).

- Provide feedback on each question (correct/incorrect) immediately after the user submits their answer.

4. Scoring and Progress Tracking

- Calculate and display the user's score at the end of each quiz.

- Track user progress by recording quiz attempts and scores.

- Allow users to view their past quiz attempts and scores.

5. Leaderboard: (optional)

- Implement a leaderboard to display top scorers for each quiz or overall.

- Rank users based on their total scores or average scores.

```
//Display top scorers for each quiz or overall
//Rank users based on total scores or overall
```

QuizService.java

```
import java.sql.*;
import java.util.List;
public class QuizService {
    private Connection conn;
    private QuizDAO quizDAO;
    private UserDAO userDAO;
       public QuizService(Connection conn) {
        this.conn = conn;
        this.quizDAO = new QuizDAO(conn);
        this.userDAO = new UserDAO(conn);
    }
    public void takeQuiz(User user, int quizId) throws SQLException {
        List<Question> questions = quizDAO.getQuestionsByQuizId(quizId);
        int score = 0;
            for (Question question : questions) {
            System.out.println(question.getQuestion());
            System.out.println("1. " + question.getOption1());
            System.out.println("2. " + question.getOption2());
            System.out.println("3. " + question.getOption3());
            System.out.println("4. " + question.getOption4());
            System.out.print("Select your answer: ");
            int userAnswer = new Scanner(System.in).nextInt();

            if (userAnswer == question.getCorrectOption()) {
                System.out.println("Correct!");
                score++;
            } else {
                System.out.println("Incorrect.");
            }
        }
        System.out.println("Your score: " + score + "/" + questions.size());                //score calculation
        saveResult(user, quizId, score);
    }
```

```
//Allow users to view past quiz attempts and score
    public void saveResult(User user, int quizId, int score) throws SQLException {
        String sql = "INSERT INTO results (user_id, quiz_id, score) VALUES (?, ?, ?)";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setInt(1, user.getId());
        stmt.setInt(2, quizId);
        stmt.setInt(3, score);
        stmt.executeUpdate();
    }
}
```

6. User Interface:

JavaFx

- Design a user-friendly interface using JavaFX, Swing or any other language as per your choice for a smooth user experience.

- Ensure clarity and consistency in the layout and navigation.

Main.java

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
 public class Main extends Application {                              //Consistent layout using JavaFx
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Online Quiz Application");

        Label label = new Label("Welcome to the Online Quiz Application");
        Button loginButton = new Button("Login");
        Button registerButton = new Button("Register");

        VBox vbox = new VBox(label, loginButton, registerButton);
        Scene scene = new Scene(vbox, 300, 200);
        primaryStage.setScene(scene);
```

```
        primaryStage.show();


        // Handle button actions and navigate to other scenes
    }
 public static void main(String[] args) {
        launch(args);
    }
}
```

7. Data Persistence:

- Use a database (e.g., SQLite, MySQL, etc.) to store user accounts, quizzes, questions, and quiz results.

- Implement CRUD operations using JDBC for database interaction.

MySQL with JDBC

DatabaseConnection.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DatabaseConnection {                    //using MySQL to Store the data
    private static final String URL = "jdbc:mysql://localhost:3306/QuizApp";
    private static final String USER = "root";
    private static final String PASSWORD = "root123";
 public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

8. Error Handling and Validation:(optional)

- Handle exceptions gracefully and provide informative error messages to users.

- Validate user input to prevent errors and ensure data integrity.

9. Security Considerations:

- Implement password hashing and salting to securely store user passwords.

Xml File

```
<dependency>
   <groupId>org.mindrot</groupId>
   <artifactId>jbcrypt</artifactId>          // For hashing and salting we can use libraries like “BCrypt”
   <version>0.4</version>
</dependency>
```

PasswordUtils.java

```
import org.mindrot.jbcrypt.BCrypt;
public class PasswordUtils {
   public static String hashPassword(String password) {
      return BCrypt.hashpw(password, BCrypt.gensalt());
   }
     public static boolean checkPassword(String plaintext, String hashed) {
      return BCrypt.checkpw(plaintext, hashed);
   }
}
```

10. Documentation: (optional)

- Provide comprehensive documentation including setup instructions, user guide, and code documentation (comments).

- Document any assumptions made and limitations of the system.

  - We can expand by adding handling exceptions.
  - Handle database connections to make app as robust and secure.

OUTPUT:

Registration

Enter username: udaykiran

Enter password: uday123

Registration successful!

Login

Enter username: udaykiran

Enter password: uday123

Login successful! Welcome,uday.

Creating a Quiz // Quiz management

Enter quiz title: General knowledge

Quiz created successfully!

Adding a Question

Enter quiz ID: 1

Enter question : What is the capital of Telangana?

Enter option 1: Dispur

Enter option 2: Panaji

Enter option 3: Hyderabad

Enter option 4: Patna

Question added successfully!

Taking a Quiz **//quiz taking**

Select a quiz to take:1

Question: What is the capital of Telangana?

1.Dispur

2.Panaji

3.Hyderabad

4.Patna

Select your answer(1-4):3

Correct!

Your Score:1/1

Viewing Score //Scoring and progress Tracking

Quiz ID:1

Score: 1

Viewing Past Attempts

Past Attempts:

Quiz ID: 1 – Score: 1

Quiz ID: 2 – Score: 0

Displaying Top Scorers //Leaderboard

Leaderboard for Quiz ID: 1

1.udaykiran – 1 point

2.udaykiran – 0 points

HERE SOME USER INTERFACE

//Main Window

[login][Register]

//Error Handling

Invalid login:

Invalid username or password.please try again.

Invalid input:

Invalid input.Please select a vaild option.

Password Hashing

Hashed password:
$2a$10$KbZB5lIlg/WmfEtM3uo5YON4Ttshn3w8ZSpCh6ZgD3AG6H19J6Gtu