```
public class User {

    String username;
    String hashedPassword;
    String salt;



    public User(String u, String h, String s) {
        this.username = u;
        this.hashedPassword = h;
        this.salt = s;
    }

}
```

```
import java.util.List;

public class Quiz {
            String title;
     List<Question> questions;

     public Quiz(String title, List<Question> questions) {
        this.title = title;
        this.questions = questions;
     }
}
```

```
public class Res {
```

```
        String username;
        String title;
        int score;

        public Res(String username, String title, int score) {
            super();
            this.username = username;
            this.title = title;
            this.score = score;
        }


}
```

```
import java.util.List;

public class Question {

    String text;
    List<String> options;
    int correctIndex;

    Question(String text, List<String> options, int correctIndex) {
        this.text = text;
        this.options = options;
        this.correctIndex = correctIndex;
    }
```

```
}
import java.util.*;
import java.security.*;
-----------------------------------------------------------------------------------------------------
public class Main {
	static Map<String, User> users = new HashMap<>();
	static Map<String, List<Quiz>> quizzesByTopic = new HashMap<>();
	static List<Res> results = new ArrayList<>();

	static Scanner scanner = new Scanner(System.in);

	public static void main(String[] args) {
		seedData();
		System.out.println("=== Welcome to the Online Quiz App ===");

		User currentUser = loginOrRegister();

		while (true) {
			System.out.println("\n1. Take Quiz");
			System.out.println("2. View Past Attempts");
			System.out.println("3. Leaderboard");
			System.out.println("4. Exit");
			System.out.print("Choose an option: ");
			String choice = scanner.nextLine().trim();

			if (choice.equals("1")) {
				takeQuiz(currentUser);
```

```
                }
                else if (choice.equals("2")) {
                        viewAttempts(currentUser);
                }
                else if (choice.equals("3")) {
                        showLeaderboard();
                }
                else if (choice.equals("4")) {
                        System.out.println("Goodbye!");
                        break;
                } else {
                        System.out.println("Invalid option.");
                }
        }
}

static User loginOrRegister() {
        while (true) {
                System.out.print("Enter username: ");
                String username = scanner.nextLine().trim();
                System.out.print("Enter password: ");
                String password = scanner.nextLine().trim();

                if (users.containsKey(username)) {
                        User user = users.get(username);
                        String hash = hashPassword(password, user.salt);
                        if (hash.equals(user.hashedPassword)) {
                                System.out.println("Login successful.");
```

```
                    return user;
                } else {
                    System.out.println("Incorrect password. Try again.");
                }
            } else {
                String salt = generateSalt();
                String hash = hashPassword(password, salt);
                User newUser = new User(username, hash, salt);
                users.put(username, newUser);
                System.out.println("User registered.");
                return newUser;
            }
        }
    }

    static String hashPassword(String password, String salt) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(salt.getBytes());
            byte[] hash = md.digest(password.getBytes());
            StringBuilder sb = new StringBuilder();
            for (byte b : hash)
                sb.append(String.format("%02x", b));
            return sb.toString();
        } catch (Exception e) {
            throw new RuntimeException("Hashing failed");
        }
    }
```

```
    public static void seedData() {
        List<Question> questions = new ArrayList<>();
        questions.add(
                new Question("What is the capital of France?",
Arrays.asList("Berlin", "Paris", "Rome", "Madrid"), 1));
        questions.add(new Question("What is 2 + 2?", Arrays.asList("3", "4", "5",
"6"), 1));
        questions.add(
                new Question("Which is the largest planet?",
Arrays.asList("Earth", "Mars", "Jupiter", "Saturn"), 2));


        quizzesByTopic.put("General", Arrays.asList(new Quiz("General Knowledge",
questions)));
    }


    public static void viewAttempts(User user) {
        System.out.println("\nYour Past Quiz Attempts:");
        boolean found = false;
        for (Res r : results) {
            if (r.username.equals(user.username)) {
                System.out.println("- " + r.title + ": " + r.score + " pts");
                found = true;
            }
        }
        if (!found)
            System.out.println("No attempts yet.");
    }
```

```
	public static void showLeaderboard() {
		System.out.println("\n=== Leaderboard ===");
		Map<String, Integer> scores = new HashMap<>();
		for (Res r : results) {
			scores.put(r.username, scores.getOrDefault(r.username, 0) + r.score);
		}

		scores.entrySet().stream().sorted((a, b) -> b.getValue() - a.getValue())
				.forEach(e -> System.out.println(e.getKey() + ": " + e.getValue()
+ " pts"));
	}

	public static String generateSalt() {
		byte[] salt = new byte[16];
		new SecureRandom().nextBytes(salt);
		StringBuilder sb = new StringBuilder();
		for (byte b : salt)
			sb.append(String.format("%02x", b));
		return sb.toString();
	}

	public static void takeQuiz(User user) {
		System.out.println("Available Topics:");
		for (String topic : quizzesByTopic.keySet()) {
			System.out.println("- " + topic);
		}

		System.out.print("Enter topic: ");
```

```
String inputTopic = scanner.nextLine().trim();

// Make topic selection case-insensitive
String matchedTopic = quizzesByTopic.keySet().stream().filter(t ->
t.equalsIgnoreCase(inputTopic)).findFirst()
        .orElse(null);

if (matchedTopic == null) {
    System.out.println("No quizzes available for that topic.");
    return;
}

List<Quiz> quizzes = quizzesByTopic.get(matchedTopic);
if (quizzes == null || quizzes.isEmpty()) {
    System.out.println("No quizzes available for that topic.");
    return;
}

Quiz quiz = quizzes.get(0); // Only one quiz per topic for now
int score = 0;

System.out.println("\nStarting Quiz: " + quiz.title);

for (Question q : quiz.questions) {
    System.out.println("\n" + q.text);
    for (int i = 0; i < q.options.size(); i++) {
        System.out.println((i + 1) + ". " + q.options.get(i));
    }
```

```
                int answer = -1;
                while (true) {
                        System.out.print("Your answer (1-" + q.options.size() + "): ");
                        try {
                                answer = Integer.parseInt(scanner.nextLine().trim());
                                if (answer >= 1 && answer <= q.options.size())
                                        break;
                        } catch (Exception ignored) {
                        }
                        System.out.println("Invalid input.");
                }

                if (answer - 1 == q.correctIndex) {
                        System.out.println("Correct!");
                        score++;
                } else {
                        System.out.println("Incorrect. Correct answer: " +
q.options.get(q.correctIndex));
                }
        }

        System.out.println("\nQuiz Complete! Your score: " + score + "/" +
quiz.questions.size());
        results.add(new Res(user.username, quiz.title, score));
    }

}
```