Quize Project

Team 15 Member

Te Sakura

Sin Panharong

Content

- Purpose
- UML
- Demo Code

Purpose

Our purpose of this project is to create a console-based quiz app with roles for Admin and Students

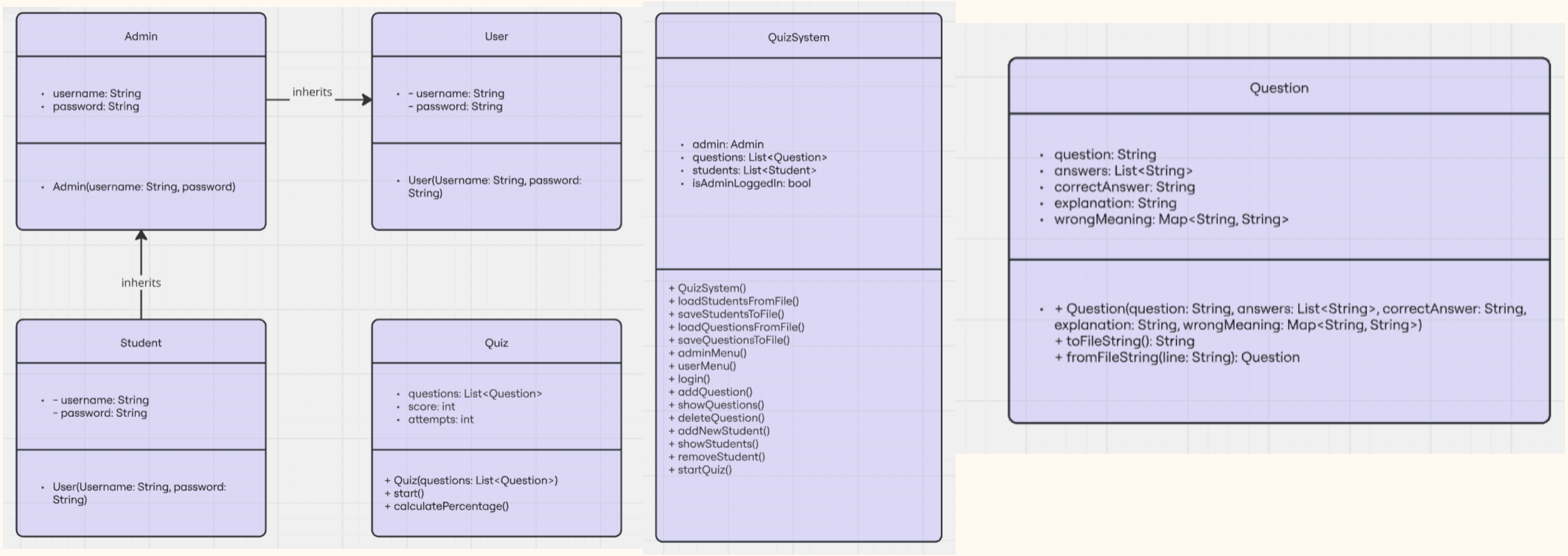
Goal

- Admin can manage questions and students.
- Students can take quizzes and receive feedback.

Features

- Admin Functions: Add/Delete Questions, Manage Students.
- Quiz: Tracks attempts and scores students.

UML


Admin
username: String
password: String
Admin(username: String, password)
inherits
User
- username: String
- password: String
User(Username: String, password: String)
inherits
Student
- username: String
- password: String
User(Username: String, password: String)
Quiz
questions: List<Question>
score: int
attempts: int
+ Quiz(questions: List<Question>)
+ start()
+ calculatePercentage()
QuizSystem
admin: Admin
questions: List<Question>
students: List<Student>
isAdminLoggedIn: bool
+ QuizSystem()
+ loadStudentsFromFile()
+ saveStudentsToFile()
+ loadQuestionsFromFile()
+ saveQuestionsToFile()
+ adminMenu()
+ userMenu()
+ login()
+ addQuestion()
+ showQuestions()
+ deleteQuestion()
+ addNewStudent()
+ showStudents()
+ removeStudent()
+ startQuiz()
Question
question: String
answers: List<String>
correctAnswer: String
explanation: String
wrongMeaning: Map<String, String>
+ Question(question: String, answers: List<String>, correctAnswer: String, explanation: String, wrongMeaning: Map<String, String>)
+ toFileString(): String
+ fromFileString(line: String): Question

User Classes

```
class User {
  String username;
  String password;

  User(this.username, this.password);
}

class Admin {
  final String username = 'admin';
  final String password = 'admin123';
}

class Student {
  String username;
  String password;

  Student(this.username, this.password);
}
```

Question class

```dart
class Question {
  String question;
  List<String> answers;
  String correctAnswer;
  String explanation;
  Map<String, String> wrongMeaning;

  Question({
    required this.question,
    required this.answers,
    required this.correctAnswer,
    required this.explanation,
    required this.wrongMeaning,
  });

  String toFileString() {
    String answersString = answers.join(",");
    String wrongMeaningString = "";
    wrongMeaning.forEach((key, value) {
      if (wrongMeaningString.isNotEmpty) {
        wrongMeaningString += ",";
      }
      wrongMeaningString += '$key:$value';
    });

    return '$question|$answersString|$correctAnswer|$explanation|$wrongMeaningString';
  }
```

Upload Data

```dart
  static Question fromFileString(String line) {
    var parts = line.split('|');
    var answers = parts[1].split(',').map((e) => e.trim()).toList();
    var correctAnswer = parts[2];
    var explanation = parts[3];
    var wrongMeaning = <String, String>{};

    if (parts.length > 4) {
      var wrongMeanings = parts[4].split(',');
      for (var wm in wrongMeanings) {
        var wmParts = wm.split(':');
        if (wmParts.length == 2) {
          wrongMeaning[wmParts[0]] = wmParts[1];
        }
      }
    }

    return Question(
      question: parts[0],
      answers: answers,
      correctAnswer: correctAnswer,
      explanation: explanation,
      wrongMeaning: wrongMeaning,
    );
  }
}
```

Fetch Data

Quiz Classes

- Display Q&A
- Analyst Wrong and Right Answer
- Track User Attempt
- Calculate the Score

```dart
class Quiz {
  List<Question> questions;
  int score = 0;
  int attempts = 0;

  Quiz(this.questions);

  void start() {
    for (var question in questions) {
      print(question.question);
      for (var answer in question.answers) {
        print(answer);
      }

      for (int attempt = 1; attempt <= 2; attempt++) {
        attempts++;
        stdout.write('Your answer (Attempt $attempt): ');
        String? userAnswer = stdin.readLineSync();

        if (userAnswer != null) {
          userAnswer = userAnswer.toLowerCase();

          if (userAnswer == question.correctAnswer.toLowerCase()) {
            print('Correct!');
            print('Meaning of the correct answer: ${question.correctAnswer} - ${question.explanation}');
            score += (3 - attempt);
            break;
          } else {
            print('Wrong!');
            String? meaning = question.wrongMeaning.entries
                .firstWhere(
                  (entry) => entry.key.toLowerCase() == userAnswer,
                  orElse: () => MapEntry('', ''),
                )
                .value;

            if (meaning.isNotEmpty) {
              print('Meaning of your answer: $meaning');
            } else {
              print('No meaning available for your answer.');
            }

            if (attempt == 2) {
              print('You have used both attempts. The correct answer is: ${question.correctAnswer}');
              print('Explanation: ${question.explanation}\n');
            }
          }
        } else {
          print('Invalid input. Please enter a valid answer.\n');
        }
      }
    }
    print('Your final score is: $score/${questions.length * 2}');
    calculatePercentage();
  }

  void calculatePercentage() {
    double percentage = (score / (questions.length * 2)) * 100;
    print('Your percentage score is: ${percentage.toStringAsFixed(2)}%');
    print('Total attempts made: $attempts');
  }
}
```

Quiz System Classes

```dart
class QuizSystem {
  Admin admin = Admin();
  List<Question> questions = []; // List to hold questions
  List<Student> students = []; // List to hold students
  bool isAdminLoggedIn = false; // Track if admin is logged in

  QuizSystem() {
    loadStudentsFromFile(); // Load students from file
    loadQuestionsFromFile(); // Load questions from file
  }

  void loadStudentsFromFile() {
    final file = File('student.txt');
    if (file.existsSync()) {
      var lines = file.readAsLinesSync();
      for (var line in lines) {
        var parts = line.split('|');
        var student = Student(parts[0], parts[1]);
        students.add(student);
      }
    }
  }

  void saveStudentsToFile() {
    final file = File('student.txt');
    var lines = students.map((student) => '${student.username}|${student.password}').toList();
    file.writeAsStringSync(lines.join('\n'));
  }

  void loadQuestionsFromFile() {
    final file = File('Q&A.txt');
    if (file.existsSync()) {
      var lines = file.readAsLinesSync();
      questions = lines.map((line) => Question.fromFileString(line)).toList();
    }
  }

  void saveQuestionsToFile() {
    final file = File('Q&A.txt');
    var lines = questions.map((q) => q.toFileString()).toList();
    file.writeAsStringSync(lines.join('\n'));
  }
```

Load Username and Password

Upload Username and password

Display Menu

```dart
void adminMenu() {
  while (isAdminLoggedIn) {
    print('\nAdmin Menu:');
    print('1. Add Question');
    print('2. Show Questions');
    print('3. Delete Question');
    print('4. Add New Student');
    print('5. Show Students');
    print('6. Remove Student');
    print('7. Log Out');
    stdout.write('Choose an option: ');
    String? choice = stdin.readLineSync();

    switch (choice) {
      case '1':
        addQuestion();
        break;
      case '2':
        showQuestions();
        break;
      case '3':
        deleteQuestion();
        break;
      case '4':
        addNewStudent();
        break;
      case '5':
        showStudents();
        break;
      case '6':
        removeStudent();
        break;
      case '7':
        isAdminLoggedIn = false;
        print('Logged out successfully.');
        break; // Just break to return to the login process
      default:
        print('Invalid choice, please try again.');
    }
  }
}

void userMenu() {
  while (!isAdminLoggedIn) {
    print('\nUser Menu:');
    print('1. Start Quiz');
    print('2. Log Out');
    stdout.write('Choose an option: ');
    String? choice = stdin.readLineSync();

    switch (choice) {
      case '1':
        startQuiz();
        break;
      case '2':
        print('Logged out successfully.');
        return; // Return to login
      default:
        print('Invalid choice, please try again.');
    }
  }
}
```

Continues

```
void login() {
  while (true) {
    stdout.write('Username: ');
    String? username = stdin.readLineSync();
    stdout.write('Password: ');
    String? password = stdin.readLineSync();

    if (username == admin.username && password == admin.password) {
      isAdminLoggedIn = true;
      print('Admin logged in successfully.');
      adminMenu();
      break;
    } else {
      bool userFound = false;
      for (var student in students) {
        if (student.username == username && student.password == password) {
          userFound = true;
          break;
        }
      }

      if (userFound) {
        print('$username logged in successfully.');
        userMenu();
        break;
      } else {
        print('Invalid login. Please check your username and password.');
      }
    }
  }
}
```

Continues

```
void addQuestion() {
  stdout.write('Enter question: ');
  String questionText = stdin.readLineSync() ?? 'No question provided';
  stdout.write('Enter multiple answers (comma separated): ');
  String? answersInput = stdin.readLineSync();
  List<String> answers = [];

  if (answersInput != null) {
    answers = answersInput.split(',').map((answer) => answer.trim()).toList();
  }

  stdout.write('Enter correct answer: ');
  String correctAnswer = stdin.readLineSync() ?? 'No correct answer';

  stdout.write('Enter explanation: ');
  String explanation = stdin.readLineSync() ?? 'No explanation provided';

  Map<String, String> wrongMeaning = {};

  for (var answer in answers) {
    if (answer.toLowerCase() != correctAnswer.toLowerCase()) {
      stdout.write('Enter meaning of wrong answer "$answer": ');
      wrongMeaning[answer] = stdin.readLineSync() ?? 'No meaning provided';
    }
  }

  var question = Question(
    question: questionText,
    answers: answers,
    correctAnswer: correctAnswer,
    explanation: explanation,
    wrongMeaning: wrongMeaning,
  );

  questions.add(question);
  saveQuestionsToFile();
  print('Question added successfully.');
}


void showQuestions() {
  if (questions.isEmpty) {
    print('No questions available.');
    return;
  }
  print('Registered Questions:');
  for (var i = 0; i < questions.length; i++) {
    var question = questions[i];
    print('Question ${i + 1}: ${question.question}');
    print('Answers: ${question.answers.join(', ')}');
    print('Correct Answer: ${question.correctAnswer}');
    print('Explanation: ${question.explanation}');

    if (question.wrongMeaning.isNotEmpty) {
      print('Wrong Meanings:');
      question.wrongMeaning.forEach((answer, meaning) {
        print(' - Answer: "$answer" means: "$meaning"');
      });
    } else {
      print('No wrong meanings available for this question.');
    }
    print('');
  }
}
```

```
void showQuestions() {
  if (questions.isEmpty) {
    print('No questions available.');
    return;
  }
  print('Registered Questions:');
  for (var i = 0; i < questions.length; i++) {
    var question = questions[i];
    print('Question ${i + 1}: ${question.question}');
    print('Answers: ${question.answers.join(', ')}');
    print('Correct Answer: ${question.correctAnswer}');
    print('Explanation: ${question.explanation}');

    if (question.wrongMeaning.isNotEmpty) {
      print('Wrong Meanings:');
      question.wrongMeaning.forEach((answer, meaning) {
        print(' - Answer: "$answer" means: "$meaning"');
      });
    } else {
      print('No wrong meanings available for this question.');
    }
    print('');
  }
}
```

```
void deleteQuestion() {
  if (questions.isEmpty) {
    print('No questions available to delete.');
    return;
  }
  showQuestions();
  stdout.write('Enter the question number to delete (starting from 1): ');
  int? index = int.tryParse(stdin.readLineSync()!);
  if (index != null && index > 0 && index <= questions.length) {
    questions.removeAt(index - 1);
    saveQuestionsToFile();
    print('Question deleted successfully.');
  } else {
    print('Invalid question number.');
  }
}
```

```
void addNewStudent() {
  while (true) {
    stdout.write('Enter new student username: ');
    String? username = stdin.readLineSync();

    stdout.write('Enter new student password: ');
    String? password = stdin.readLineSync();

    if (username == 'admin') {
      print('Sorry, this username is reserved. Please choose another one.');
      continue;
    }

    bool isUsernameTaken = false;
    for (var student in students) {
      if (student.username == username) {
        isUsernameTaken = true;
        break;
      }
    }

    if (isUsernameTaken) {
      print('Sorry, this username is already taken. Please choose another one.');
    } else {
      if (username != null && password != null) {
        students.add(Student(username, password));
        saveStudentsToFile();
        print('New student added: Username - $username');
        break;
      }
    }
  }
}
```

Continues

```dart
void removeStudent() {
  if (students.isEmpty) {
    print('No students available to remove.');
    return;
  }
  showStudents();
  stdout.write('Enter the username of the student to remove: ');
  String? username = stdin.readLineSync();

  var studentToRemove = students.firstWhere(
    (student) => student.username == username,
    orElse: () => Student('', ''),
  );

  if (studentToRemove.username.isNotEmpty) {
    students.remove(studentToRemove);
    saveStudentsToFile();
    print('Student "$username" removed successfully.');
  } else {
    print('Student not found.');
  }
}
```

```dart
void showStudents() {
  if (students.isEmpty) {
    print('No students available.');
    return;
  }
  print('Registered Students:');
  for (var student in students) {
    print('- ${student.username}');
  }
}
```

```dart
void startQuiz() {
  if (questions.isEmpty) {
    print('No questions available for the quiz. Please ask the admin to add questions.');
    return;
  }
  Quiz quiz = Quiz(questions);
  quiz.start();
}
```

Demo Time