QUIZ APP

Jalol Mirislomov

Introduction

ABOUT

Console based quiz application built on java 19. App has login and registration feature and it allows only unique username. App has leaderboard which displays 10 best finished quizzes. App uses the MVC design pattern, and it has validation for inputs and services to validate user authentication. App has 2 type of users: admin, simple user. Admins have access to the admin panel, which allows them to edit and manage questions, options for question, users, categories and they can delete finished quizzes from database. App has all basic functions that other quizzes have.

KEY FEATURES

01

Auth

App automatically displays information depends on user type. For example: if user is admin, he can see admin panel option on main menu. However, simple users can't.

02

Validation and other services

App has input validator. Service that checks is user auth or is user admin and more…

03

MVC design pattern

MVC (model, view, controller) pattern which provides a scalable and maintainable architecture.

04

CRUD

App uses CRUD function for all models, which allows easily management and editing of data

DATABASE

I have used CSV (comma-separated values) files as database. There are data classes which has connection with CSV files. What they do are: reading from file, writing to file, update from file. Also they include algorithm that generate unique ID for each data table (like auto incrementing on sql databases). All data classes have interface. Reason creating interfaces for data classes is in the future, app can switch to real database like mysql, postgresql and interfaces helps easily create function that needed.

DATA TABLES

Here is some examples of data base structures.

I designed databases myself, there can be wrong use cases or better solutions.

Type	Users
int	id
String	name
String	login
String	password
boolean	is_Admin

Type	Quizzes
int	id
int	user_id
int	correct_aswers
int	size_questions
double	score
String	created_at

Type	Questions
int	id
int	category_id
String	question
String	answer
String(with commas)	options

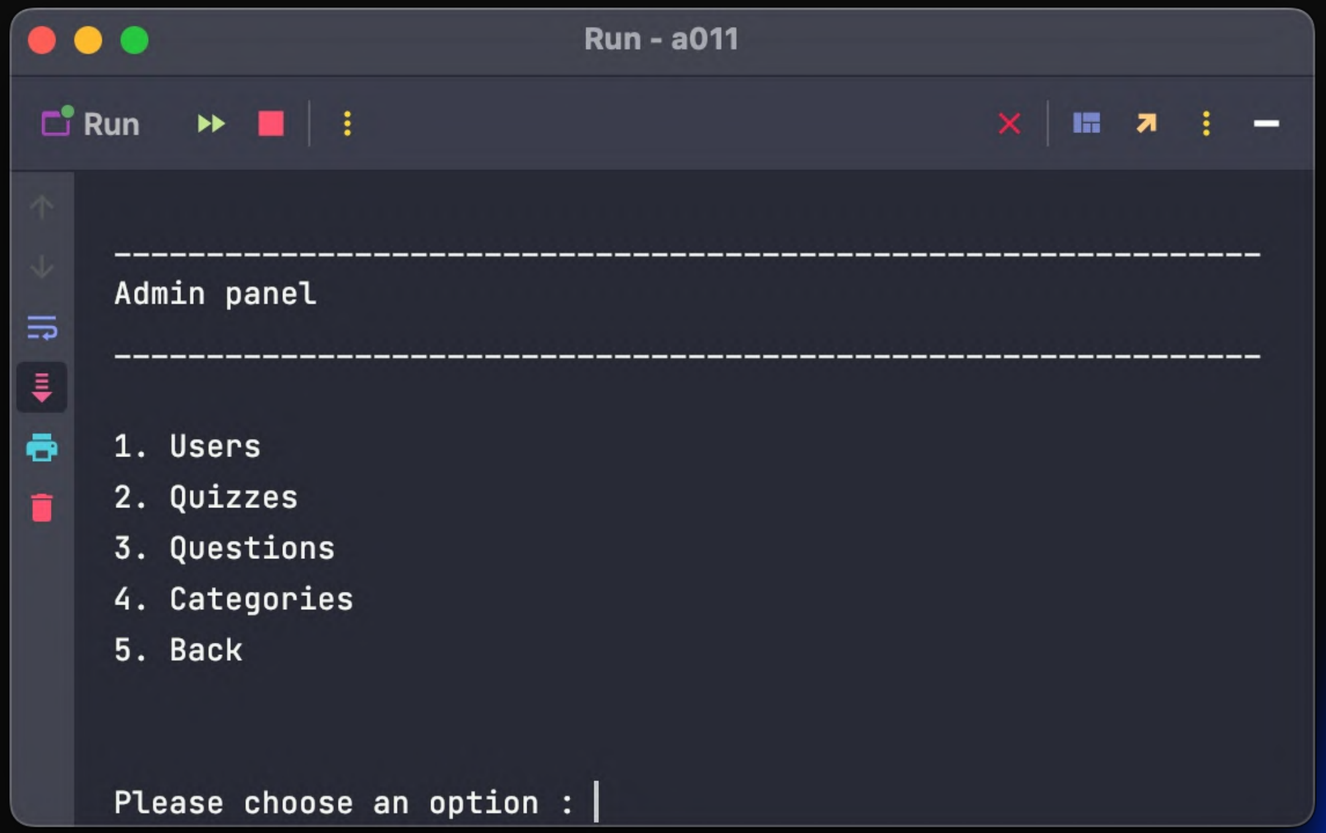
UNIT TESTS

Example code of register unit test

```
package Tests;

import Data.UserData;
import Models.User;
import Views.Messenger;

import java.sql.Timestamp;
import java.text.SimpleDateFormat;

class AuthControllerTest {

    @org.junit.jupiter.api.Test
    void register() {
        UserData userData = new UserData();
        int id = userData.getUpdatedId();
        String name = "Test name";
        SimpleDateFormat timestamp = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss.SSSXXX");
        String login = "testLogin"
        + timestamp.format(new Timestamp(System.currentTimeMillis()));
        String password = "testPassword";
        boolean isAdmin = false;
        if (userData.isUserNameUnique(login)) {
            Messenger messenger = new Messenger();
            messenger.oneLineTitle("Username is already taken!");
            return;
        }
        User user = new User(id, name, login, password, isAdmin);
        userData.saveUser(user);
    }
}
```

VIEWS

Example screenshot of admin panel



IDEAS FOR FUTURE

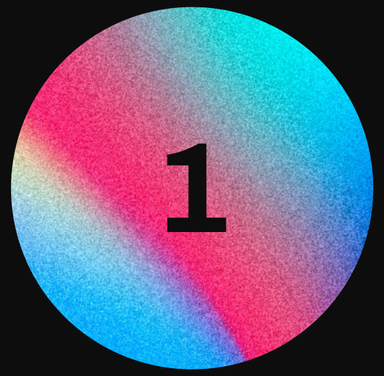
Here is my ideas for future improvement

Avoiding using CSV files as database. Instead of it use SQL database.

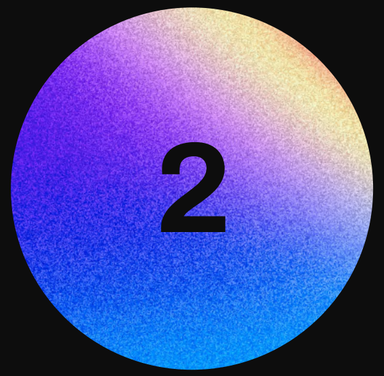
Hashing passwords. If app is hacked, cybercriminals can't get access to your password.

Make UI using some library like GUI. It makes app much understandable.

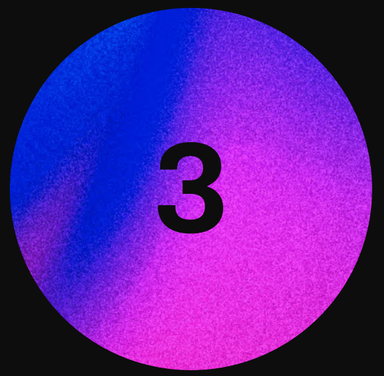
GET STARTED



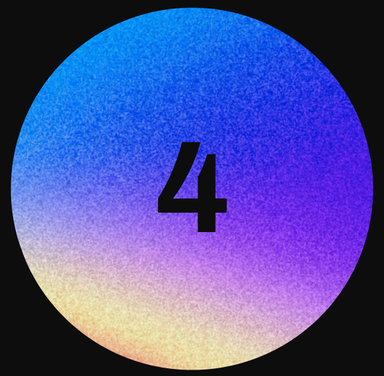
Open the project in Intellij idea. Set up your JDK.



Try to run main file.



If you have issue with junit.jupiter.api package, Add library 'JUnit5.8.1'.
Other case, delete tests package.



Choose login option.
username: admin,
password: admin.

I manually set up admin account on CSV file.

CONTACT OR FEEDBACK

If you have any questions or you want to give some feedback to this project - feel free to contact me

Personal email:
mirislomovmirjalol@gmail.com

Student email:
mm2456@student.aru.ac.uk

Github account
https://github.com/mirislomovmirjalol