Test Report Quiz application

Introduction

Applications need to be tested to make sure they work properly. Therefore our quiz application is tested as well. Different parts of our application are tested in different ways. Our backend is tested using automated Junit tests while our frontend has been tested manually. In this document the different testing methods will be discussed.

Backend testing

The backend of our application includes all the server side code and utility methods of our application. These methods are straightforward to test using junit tests, as they receive a select input and always output the same result. Methods are tested to check if they give the correct output using some specified inputs and if they throw correct and clear errors when they don't receive the correct input or something went wrong during execution. We also tested our pseudo random question generators using Junit tests. These methods don't give the same output every time, but they get tested if the output that is given is in the correct format and usable by the rest of our application. These methods get tested multiple times by the same tests during an application test to account for the randomness that these methods produce.

Frontend testing

Why no automated tests

The frontend and ui interface has been tested manually. We could not test this part automatically with Junit tests, because a user can do a lot of different things with a mouse and keyboard. A user can click a button one time or multiple times in a row. Also a user can type text in a number field. These kinds of inputs all need to be tested and as there are so many different possibilities, this is not feasible using Junit tests. Therefore we decided to test this part of our application manually. The next section will explain how we did this.

How we test our application manually

When testing our application manually, you need to start up both a server and at least one client. First we start using our application like a person is supposed to use it. So with patience and filling fields with the values you are supposed to fill. When we have concluded that the application functions how it is supposed to function when used normally, we put more pressure on the application. We use it impatiently and click buttons multiple times in a row. Also we fill in text where a user is supposed to fill in numbers, and we look at how the application responds to this kind of behavior. Another thing we do to test our multiplayer part, is connecting multiple clients to a game and then closing some clients with the task manager. This way the server will not receive a message from the client that it is disconnecting so we can see what happens to other clients.

Conclusion

Using these techniques we have tested our application on errors and fixed a lot of bugs. When our application is expanded and more features are added, these will also be tested using the same techniques discussed earlier, Junit tests for backend code and manual testing for the frontend of our application. This way our application is fully tested and most bugs are discovered.