Programing\_VS\_15605

**Introduction:**

The test plan’s purpose is to test a majority of the important functions of Programing\_VS. This means that the project in question will be able to at least be functional while working within my constraints. The constraints of this project is time and money due to being a college student.

**References:**

[Project Proposal](https://github.com/kilikwhite/CSU-Senior-Project/blob/master/docs/Proposal.md)

[Requirement Document](https://github.com/kilikwhite/CSU-Senior-Project/blob/master/docs/Requirements%20Specification.docx)

[My Final Project Repository on GitHub](https://github.com/kilikwhite/CSU-Senior-Project)

**Test Items:**

The test Items that are going to be used for the test are 2 windows computers with the software installed and functioning. These computers will help with testing both the front end and the back end of the project.

**Features to be Tested:**

The features that are going to be tested are, the practice mode, the select lobby connection type, the question selector, if the code compiles in both practice mode and in a lobby, if the test file for the lobby connection mode works, and if the time the other player is done shows up on the other person’s screen.

**Features Not to Be Tested:**

The features that will not be tested are compatibility with apple and mobile devices since the project is not for those devices. Next is the security features because security is not a priority for this project, but I am going to find a way to hide the IP addresses for the project.

**Approach:**

My overall approach to testing will be more reactive since I will try to fix the errors as the programming continues. As for what testing system is going to be used, I will use the Functional Testing route. The route includes unit testing, integration testing, and system testing, with the method being Gray Box testing.

**Item Pass/Fail Criteria:**

The criteria that will be used is if the unit tests, integration tests, and system tests all have an 85 percent passing rate.

**Test Deliverables:**

| Test Case: | Expected Result: | Actual Result: |
| --- | --- | --- |
| Check the response on pressing the ‘practice button’ | The screen will shift to a practice page where there will be a question and the Ace Editor active |  |
| Check the response on pressing the ‘create a lobby button’ | The screen will shift to another page with three buttons named, ‘Create Spectator Only Lobby’, ‘Create Regular Lobby’ and ‘Join Lobby’ along with a form page that asks for a Screen Name and a room id |  |
| Check if the inputs from the Screen Name textbox and Room ID textbox got sent to the client page by any of the lobby related buttons | The info from the textboxes well be sent into the client side of the program and later to the server. |  |
| Check the response on pressing the ‘Create Spectator Only Lobby’ button | Assuming the inputs for Screen Name and Room ID are correct. It should send the user to a screen with the Screen Name and Room ID being at the right-hand top corner. |  |
| Only in the lobbies check if the Screen Name and Room ID show up on the right-hand  top corner. | The Screen Name and Room ID will show up on the right-hand top corner in bold |  |
| Check the response on pressing the ‘Create Regular Lobby’ button | Assuming the inputs for Screen Name and Room ID are correct. The user will be placed into a room where the Room ID and Screen Name is shown in the top right-hand corner. The Ace text editor formats will show up on the page as well. |  |
| Check the response on pressing the ‘Join Regular Lobby’ button | First the Room ID has to match a current Room ID which should be true. The user will be placed into a room where the Room ID and Screen Name is shown in the top right-hand corner. The Ace text editor formats will show up on the page as well. |  |
| Check if the text inputted in the input section for practice and lobby modes will have a color change for the text. | When typing a variable or a method in the input section, it will be a different color depending on what was typed |  |
| Check if pressing the run button will compile the code in the input section | The code in the input section will compile and will return ‘Test # passed’ or ‘Test # failed’. |  |
| Check if the compiled code will return the results in the output section | The compiled code will show at a section as either ‘Test # passed’ or ‘Test # failed’. |  |
| Check if the incorrect code in the input section will compile and return ‘Test # failed’ and depending on # the actual value | The output section will show ‘Test # failed’ and an error message depending on # |  |
| Check if in any lobby mode, it displays a timer | After the lobby was made, a timer will be in the top left of the screen |  |
| Check response of the Log menu dropdown in a Regular Lobby. | When pressing the Log menu button, a dropdown will show any recent presses of the run button by a user both yourself and another person in the room along with how many test cases passed, and your timer at the time of pressing the run button. |  |
| Check if someone in a Regular Lobby mode compiles code, the other person can see when they compiled the code and how many Test Cases passed if in the same room after pressing run. | After the run button is pressed, the Log menu dropdown will populate with the user’s Screen Name, # of passed Test Cases, and the timer at which the button was passed. |  |
| Check the response of the Log section in the Spectator Lobby | The Log section will show nothing unless a user from a regular lobby mode is inside of the spectator lobby |  |
| Check if someone in a Regular Lobby mode compiles code, the other person in a spectator lobby can see when they compiled the code, how many Test Cases passed, the person’s Screen Name, and the code itself if in the same room after pressing run. | The Log section will populate with the user’s Screen Name, the user’s Code, the # of passed Test Cases, and the timer at which the button was pressed |  |

**Test Environment:**

Since I am going to use both my home computer and my laptop for testing along with my advisor’s machine for more testing. I will specify the info for each:

Kilik’s Home Computer

OS: Windows 10

Processor: AMD Ryzen 7

RAM: 32.0 GB

System type: 64-bit operating system

Node.js version: 16.10.0

Kilik’s Laptop:

OS: Windows 10

Processor: Intel Core i5

RAM: 12 GB

System type: 64-bit operating system

Node.js version: 17.3.1

The software I am using is the code editor VS code to make the project along with using meteor’s tests with it.

**Estimate:**

The test will cost a considerable amount of time for the developer since the dev is working by himself on the project.

[**Schedule:**](https://docs.google.com/spreadsheets/d/1D9RFGPq201_ZofBapEdoPFSa6vUdPikjnpKLe370pNs/edit?usp=sharing)

**Responsibilities:**

Developer/Tester:

The Developer is responsible for programming the entire project and making sure there are no compilation errors. Since the single developer is also the tester, the tester also has to make sure that there are no errors in running the application and the app runs smoothly.

**Approvals:**

Kilik R. White(role: the creator/programer of the project)

X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Hayes(role: the overseer of the project)

X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_