|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test NO.** | **Description** | **Expected Result** | **Actual Result** | **feedback** | **Improvements** |
| 1 | Start-up Screen | When the program is run it should pop up with the buttons and background present | All buttons appeared and background was present | None | None – Done as individual implementation by group member |
| 2 | Play button | Hovering over it should shadow the button and clicking it should load up main window and play music | Button effect was present. Clicking the button brought up main window and music was played | None | None - Done as individual implementation by group member |
| 3 | User input for first question | Program accepts users response | Program did not accept different ways of entering a colour such as Red ReD RED reD ect. | The feedback we got here was that different users might enter the colours in different ways so we needed to inherit these. | We made it so that the program would accept all variations of ways of a word. |
| 4 | User input for second question | Label expected to change after user has input the answer | Label did change, but again different variations of the word was not accepted.  Eg. NW nw Nw nW | Feedback here was that different variations of a word were not recognised | We made it so that the program would accept all variations of ways of a word |
| 5 | User input for third question | User should not be able to enter more than 14 items to search for | User was able to search for more than 14 items and program also accepted negative numbers | We did not have any validation here just like previous user input scenarios. | We added validation so that the program would not accept items more than 14 and would not accept negative values. |
| 6 | User input for countdown | A large countdown or 0 input should not be accepted. | A large countdown and a 0 entry was accepted. | We again did not have any user constraints in place to prevent these mistakes. | We implemented validation so that the user cannot have a large countdown timer and 0 or minus numbers were not accepted. |
| 7 | Robot movement | Robot should move smoothly around the program interface | Robot was able to smoothly slide across the screen and collect the items. | None | None |
| 8 | Robot collision | Should avoid going through coloured squares which haven’t been chosen by the user | The robot went through coloured squares which the user didn’t specify and made them vanish after it has passed over them. | The reason this could be is because we originally used arrays to store the different elements and values within our program. | This was fixed by implementing an SQL database into the python GUI. This was done by a group member as their individual implementation. |
| 9 | Item collection list shown | Once the number of items is collected which was specified by the user or the timer runs out; should show list of items from cheap to pricey. | Once the number of items is collected which was specified by the user or the timer runs out; showed show list of items from cheap to pricey. | None | None |
| 10 | Quit button on start up | When the button is clicked the program should be stopped and closed. | Button was clicked, the program was closed | None | None - Done as individual implementation by group member |
| 11 | [X] Button | Should kill program | Killed program | None | None |