Validation Testing

Table of Contents

Validation Testing

This collection of test cases is meant to check if we have fulfilled the functional requirements for the program at a high level. They are validating that we have met the required functions for the application. To accomplish this we have to manually check through the screens of the application to ensure that we have all the needed functions. As the level of abstraction for these tests is high, we have to manually check the requirements. We cannot automate these tasks as they are abstracted from the underlying code, thus we cannot use JUnit tests to verify that we have met all the requirements. Each of these test cases verifies that we have met an individual functional requirement. They are meant to check that the overall code implements everything it is meant to. Testing for the non-functional requirements is more subjective and thus was not included in the validation testing. To test the non-functional requirements we would have to rely on feedback from the users of the game. We cannot test if our game is easy to use and educational. We would need to have the target audience use it and provide feedback to test these things. We have done our best to implement the non-functional requirements as we implemented each of these functional requirements. But, due to their subjective nature, we did not find it appropriate to have tests to verify their implementation. Thus, the tests focus only on verifying that we have implemented each of the functional requirements outlined in the requirements documentation.

Test Case Name	GUI Mouse responsiveness
Test Case Description	This test case is meant to test if the GUI responds to mouse inputs given by the user.
Test Steps	1. Log in to the game. 2. Navigate through the GUI using the mouse as the input device. 3. Observe the response of the program to mouse inputs.
Pre-Requisites	The program must be booted up to be able to verify that the GUI responds to mouse inputs. The user must have an account to access the program.
Expected Results	As the user uses the mouse to provide input to the program, the program will respond to the inputs. This can manifest in a few different ways. The program may make a sound to make it obvious that the button has been pressed. Alternatively, the screen of the GUI might change as the user navigates through the various screens.
Test Category	Validation Testing
Requirement	Functional requirement 1. Graphical user interface 2. The GUI is mouse-based so that the user can click on buttons/ text boxes to navigate the different menus and progress through gameplay.
Automation	Manual
Date Run	30 Mar 2024
Pass /Fail	Pass
Test Results	The GUI responds to mouse inputs.
Remarks	The GUI smoothly responds to mouse inputs. This function works as intended.

Test Case Name	GUI Keyboard Responsiveness
Test Case Description	This case tests if the GUI responds to keyboard inputs.

Test Steps	1. Log in to the game. 2. Navigate through the GUI using keyboard inputs 2. Observe if the response from the GUI is as intended.
Pre-Requisites	The program must be booted up to be able to verify that the GUI responds to keyboard inputs. The user must have an account to access the program.
Expected Results	As the user uses the keyboard to provide input to the program, the program will respond to the inputs. This can manifest in a few different ways. The program may make a sound to make it obvious that the button has been pressed. Alternatively, the screen of the GUI might change as the user navigates through the various screens.
Test Category	Validation Testing
Requirement	Functional requirement 1. Graphical user interface 2. There are alternative keyboard shortcuts that allow the user to access different menus/ features directly.
Automation	Manual
Date Run	30 Mar 2024
Pass /Fail	Fail
Test Results	The GUI does not respond to keyboard inputs.
Remarks	We ran out of time to implement keyboard inputs, the program does not respond to them as originally intended.

Test Case Name	GUI Screens Test
Test Case Description	This test is to check that the GUI has all the desired screens included. These elements also have to be easily accessible. It must be obvious how to navigate through the different screens.
Test Steps	1. Log in to the game. 2. Navigate through the GUI using mouse inputs 3. Ensure that all the desired screens are accessible and it is obvious how to get to them. 4. Ensure that the back button works on each screen as they are navigated through.
Pre-Requisites	The program must be booted up. The user must have an account to access the program.
Expected Results	All desired screens will be in the GUI. Each of them will be easy to find, and easy to return to the main menu from.
Test Category	Validation Testing
Requirement	Functional Requirement Graphical User Interface
Automation	Manual
Date Run	30 Mar 2024
Pass/Fail	Pass
Test Results	The GUI includes all the specified screens and navigation between them is smooth.
Remarks	The GUI looks as intended.

Test Case Name	Main Menu Elements Test

Test Case Description	This test case checks that all the desired elements are included in the main menu.
Test Steps	1. Login to the game 2. Check that all the desired elements are included in the main menu. 3. These include name, logo, wallpaper, button to play the game, button to open the instructions, button to open the high-score screen, button to access the progression screen, button to open the settings, button to exit the game, and information about who created the game.
Pre-Requisites	Boot the program up. The user must have an account to access the program.
Expected Results	All the listed elements are included in the main menu. They are all visible in the GUI.
Test Category	Validation Testing
Requirement	Functional Requirement Graphical User Interface Main Menu (Screen)
Automation	Manual
Date Run	29 Mar 2024
Pass/Fail	Pass
Test Results	The main menu includes all the specified elements.
Remarks	The menu has all the listed elements and there are custom graphics for them. The created art is properly shown in the GUI.

Test Case Name	Tutorial Screen Elements Test
Test Case Description	This test case checks that all the desired elements are included in the tutorial screen.
Test Steps	1. Log in to the game. 2. Navigate to the tutorial screen from the main menu screen. 3. Check that the tutorial screen has the following elements; a button to return to the previous screen, a list of all mouse inputs, tips on how to interact with the gameplay screen, and a summary of how the game works.
Pre-Requisites	The program must be booted up. The user must have an account.
Expected Results	The user will be able to navigate to the tutorial screen from the main menu. In the tutorial screen, they will see all the elements listed above. The screen will give a concise overview of how to use the program and play the game. They will be able to seamlessly navigate back to the main menu using the back button.
Test Category	Validation Testing
Requirement	Functional Requirement
Automation	Manual
Date Run	31 Mar 2024
Pass /Fail	Pass

Test Results	The tutorial screen contains all the specified elements.
Remarks	The screen gives a concise overview of how to use the application. It is effective at teaching the users how to use the application and play the game.

Test Case Name	Save Game Test
Test Case Description	Test that the game can save progression to a CSV file stored in a local folder.
Test Steps	1. Login to or register an account. 2. Navigate to the gameplay screen from the main menu. 3. Play through the first level. 4. Save the game on the 2nd screen.
Pre-Requisites	The folder to store the CSV files has been created on the machine. The user can create an account and log in to the pre-existing account after. The program can create a CSV file to store the user data and the progression data.
Expected Results	The user can store their progression in the game. This is to allow the user to load back into their save at a later date. This will also allow the viewing of their progression on the progression screen.
Test Category	Validation Testing
Requirement	Functional Requirement Save And Load
Automation	Manual
Date Run	30 Mar 2024
Pass/Fail	Pass
Test Results	The program creates a CSV file and stores the data as expected.
Remarks	The saving of files works as intended.

Test Case Name	Load Game Test
Test Case Description	Test that the game can load progression from a CSV file stored in a local folder
Test Steps	1. Login to or register an account. 2. Press the continue/play button on the main screen 3. Verify that the game has resumed at the point it was left, and hasn't gone back to the beginning.
Pre-Requisites	The folder to store the CSV files has been created on the machine. The user can create an account and log in to the pre-existing account after. The program can create a CSV file to store the user data and the progression data.
Expected Results	The user can resume the game where they left off previously.
Test Category	Validation Testing
Requirement	Functional Requirement Save And Load
Automation	Manual
Date Run	30 Mar 2024
Pass/Fail	Pass
Test Results	The user can load their previously saved games and resume where they left off.
Remarks	The loading of files works as intended.

Test Case Name	High-Score Table Screen Test
Test Case Description	This test is to test that the integration of the high score table has been successful
Test Steps	1. Login to or register an account. 2. Navigate to the high score screen from the main menu by pressing the high score button. 3. Verify that the correct elements are present on the screen. 4. Verify that it is possible to navigate back to the main menu.
Pre-Requisites	The program must be booted up. The user must have an account.
Expected Results	The user sees the following elements: the back button, the table shows the 5 highest scores in descending order, and each entry shows the name of the player and the score they achieved.
Test Category	Validation Testing
Requirement	Functional Requirement High Score Table (Screen)
Automation	Manual
Date Run	30 Mar 2024
Pass/Fail	Pass
Test Results	The screen contains all the desired elements and the back button works properly.
Remarks	The high score screen works as intended.

Test Case Name	Login Screen Test
Test Case Description	This test is to test that the integration of the Login Screen has been successful.
Test Steps	1. Log in to or register an account after booting up the game. 2. Ensure that both functions work. Either the user can use their already created account or make a new one.
Pre-Requisites	The program must be booted up.
Expected Results	The user can either log in to their pre-existing account or create a new one based on their needs. A file is either read or created to facilitate this.
Test Category	Validation Testing
Requirement	Functional Requirement Multi-User Login (Screen)
Automation	Manual
Date Run	30 Mar 2024
Pass/Fail	Pass
Test Results	The user can access the login screen as it pops up when the game is booted. The user is then able to either register an account or log in to their existing account, as intended.
Remarks	The login screen works as intended.

Test Case Name	Teacher Mode Test
Test Case Description	This test is to test that the integration of Teacher Mode has been successful.

Test Steps	1. Log in to the program using the username "teacher" and the secret password. 2. Navigate to the progression screen through the main menu. 3. Verify that the progression screen is different for the teacher, and they can view all the student's progression. 4. Navigate back to the main menu. 5. Start a game and verify that the teacher can access any level in the game regardless of whether they have played through it already.
Pre-Requisites	The application has been launched. A special login for the teacher has been created with special permissions different from the normal student permissions.
Expected Results	The teacher can log in without having to create an account as they can use the special pre-existing account to access the game. After the teacher logs in they can access the progression screen and view the progression of all their students. They also can play any level in the game and check to ensure that the level is possible for their students to beat.
Test Category	Validation Testing
Requirement	Functional Requirement Teacher Mode
Automation	Manual
Date Run	31 Mar 2024
Pass /Fail	Pass
Test Results	We can log into the teacher's account and see the student's progression. We are also able to play any level we desire.
Remarks	Teacher mode works as intended and has all the required functions.

Test Case Name	Developer Mode Test
Test Case Description	This test is to test that the integration of Developer Mode has been successful.
Test Steps	1. Log in to the game using the username "developer" and a secret password 2. Access the crash reports from the main menu and verify that they're accurate 3. Access the debug menu and print the action chain in an encoded string format
Pre-Requisites	The application has been launched. A special login for the developer has been created with special permissions different from the normal student permissions.
Expected Results	The developer can log in through a unique username and password. They are then able to access a crash report generated by the program. They are also able to access a debug menu which shows the action chain from the gameplay menu printed out in a String format.
Test Category	Validation Testing
Requirement	Functional Requirement Developer Mode
Automation	Manual
Date Run	31 Mar 2024
Pass/Fail	Pass
Test Results	The user can log in as a developer and access the desired features. They are also able to access all the features of the teacher and student. Allowing them to use the program like any other user, just with some extra permissions.
Remarks	Developer mode has been implemented and works as desired.

Test Case Name	Error Handling Test
Test Case Description	This test is to ensure that the program can handle incorrect inputs and not crash.
Test Steps	1. Log in to the game 2. Try to input commands that are not correct. Observe what happens. 3. Use the app in a way that it was not intended to try and crash it. Observe if any crashes occur. 4. Close the app and open Task Manager to see if any background processes remain.
Pre-Requisites	The application has been launched. The user has an account to log into.
Expected Results	The app tells the user that the inputs are incorrect. The incorrect inputs do not cause the app to crash. It is impossible to test for every possible case that may cause the app to crash, but for the ones we do test the app should not crash. Just return to the main menu. When the app is closed, there should not be any remaining processes going on in the background.
Test Category	Validation Testing
Requirement	Functional Requirement Error Handling
Automation	Manual
Date Run	30 Mar 2024
Pass /Fail	Pass
Test Results	The app can handle the incorrect inputs we tested without crashing. There are no background processes left over when the app is closed.
Remarks	The app can handle errors as intended.

Test Case Name	Progression Screen Test
Test Case Description	This test it to ensure the progression screen properly shows the student's progression through the game.
Test Steps	1. Log in to the game to an account that already has some amount of progress through the levels. 2. Navigate to the progression screen from the main menu. 3. Ensure that the screen shows the progression through the game achieved by the student.
Pre-Requisites	The program is booted up. There is an account with some level of progression through the game.
Expected Results	The user can see the progression through the game that they have previously achieved.
Test Category	Validation Testing
Requirement	Functional Requirement Progression (Screen)
Automation	Manual
Date Run	30 Mar 2024
Pass/Fail	Pass
Test Results	The progression through the levels is properly displayed on the progression screen. This will also allow it to be seen in teacher mode.
Remarks	The progression screen is implemented and working as intended.

Test Case Name	Pause Menu Test
Test Case Description	This test is to ensure that there is a pause menu accessible during the gameplay.
Test Steps	1. Log in to the program. 2. Start a new game. 3. Pause the game after the game has been started. 4. Open the settings screen. Ensure it has the desired elements. 5. Navigate back to the pause menu. 6. Open the tutorial screen. Ensure it has the desired elements. 7. Save the game from the pause menu. 8. Navigate back to the main menu.
Pre-Requisites	The game is booted up. The user has an account.
Expected Results	The user can access all the mentioned screens. The screens are fully functional and the user can navigate back to the pause menu from the screens. The user can save their progress so that they can resume their run. The user can easily navigate back to the main menu from the pause screen.
Test Category	Validation Testing
Requirement	Functional Requirement Pause (Menu)
Automation	Manual
Date Run	30 Mar 2024
Pass/Fail	Pass
Test Results	The screens are all implemented and functional. The save game function works and we can navigate back to the main menu.
Remarks	The pause menu is functioning as desired.

Test Case Name	Settings Menu Test
Test Case Description	This test is to ensure the settings menu works and can change the settings.
Test Steps	1. Log in to the game. 2. Navigate to the settings menu from the main menu. 3. Change the colorblind mode and ensure that it changes. 4. Change the volume and ensure that it changes. 5. Navigate back to the main menu.
Pre-Requisites	The program is booted up. The user has an account.
Expected Results	The user can change the above settings and notice a difference in the application. The user can easily navigate back to the main menu from the settings menu,.
Test Category	Validation Testing
Requirement	Functional Requirement Settings (Menu)
Automation	Manual
Date Run	30 Mar 2024
Pass/Fail	Pass

Test Results	It is possible to adjust the settings in the menu as intended. The navigation back to the main menu is seamless. The menu is functional.
Remarks	The settings menu is implemented and works as intended.

Test Case Name	
Test Case Description	
Test Steps	
Pre-Requisites	
Expected Results	
Test Category	
Requirement	
Automation	
Date Run	
Pass/Fail	
Test Results	
Remarks	