**Testing Documentation**

Test 1: ReadColourDetailsFromFile

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test is used to read the colour config file in order to make sure it was formatted correctly.

The expected result for this test is that each line of the text file was looked at and checked if it contained a valid value then will return a value and the test would pass if the value wasn’t null.

Test 2: WriteColourDetailsToFile

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test is used to update the the colour config file with new values in order to test that the script which causes the colour values in the Web App to change if functioning properly.

The expected value is for the WriteColoursToFile function to return true.

Test 3: DatabaseAccessTest

This is a basic test in order to check the foundation on how the C# code will send SQL commands to the database. This test will just perform a simple “Select \*” on one of the database tables.

The expected result is for the returned list of tables to not be null. As otherwise it would of failed.

Test 4: ReadJoinCodeFromDatabase

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test is used to get the join code for a test through the use of a SQL query so the correct database table can be accessed when creating or joining a test.

The expected results for this test is that returned list was not null and contained the join code.

Test 5: AddNewQuizInfoTest

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test is used to create a new test information which enters the name of the test and the test type in order to test the code which the “Create Quiz” page will use.

The expected results for this test is that it will return a “JoinCode” once completed and the tst checks that join code isn’t null.

Test 6: AddNewQuizTableTest

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test is used to initially set up the quiz creation screen and to give the answers all default values.

The expected outcome for the test is that the “Join Number” is returned in order to show the SQL command ran successfully.

Test 7: UpdateCurrentQuizDatabase

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test was used in order to make sure that the quizzes info was updated in the central quiz database.

The Expected outcome for this test is that the function returns true to make sure the SQL statement ran successfully.

Test 8: UpdateCurrentQuizQuestion

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test is used to test out the code when saving changes to a quiz question in order to make sure it was working properly before creating the Web App page.

The Expected outcome for this test is that the function returns true to make sure the SQL statement ran successfully.

Test 9: AddNewQuizTableTestAndAddNewRow

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test was used to make sure the function of adding new questions to an existing quiz before it was implemented onto the Web App itself.

The Expected outcome for this test is that the function returns true to make sure the SQL statement ran successfully.

Test 10: GetIDValueFromCurrentQuizTest

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test was used in order to get the ID Value number for a question number in the quiz database. This was needed because when deleting a question it would mess up the positioning and could cause an error so this was used to make sure the quiz numbers are ordered sequentially.

The expected results for this test is that the Database query will return a list of all the ID values used for the quiz questions.

Test 11: UpdateJoinQuizDatabase

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test was used when developing the “Join Quiz” section of the Web App. The goal of this test is to update the “AnwerQuiz” database to contain the current Quiz info that is being answered.

The expected results for this test if for the “WriteToJoinQuiz” function to return true showing that the SQL statement has passed.

Test 12: PasswordChangeTest

This test was used when as part of the test driven development which I used when designing the back end part of the Web App.

This test was used for testing the SHA-256 hashing for passwords as well as used as an automated way for hashing and adding the passwords to the database which wasn’t an intended feature for the web app but just to help with development.

The expected results for this test is that the “AddPasswordToDatabase” function to return true showing that the SQL statement has passed.