­­­G51FSE Assessed Lab 4

|  |  |  |
| --- | --- | --- |
| **The Assemblers with Flip-Flops** |  | 09/03/2018 |

# 3. Test Plans

This section of the software specification document highlights the details of the system through the use of Unified Modelling Language and prototypes.

We have added some additional columns to the bug table. These are notes and a description for the changes are given in this paragrah.We have decided to have additional columns in the test table so we can provide more detail about the tests carried out. This will increase the traceability of the tests and make the changes easier to see. Firstly, we added **a Test ID** so that we can have a unique id for each test carried out. This will make it easier to refer back to the tests later on in the document. In addition to this, we added a column for prerequisite which will store detail about what other functions need to be performed before this test is carried out. Next, we included a column for the **actual output**. This is necessary so that we can compare if the expected output matches the actual output. If it doesn’t, a reason must be provided as to why and most likely there is an error/flow in the program. In addition to this, we added a column called ‘**test created by’**. This will hold the name of the person who created the tests and carried it out. This will increase accountability. Next, we added a column called ‘**pass or fail’**. This will make it easier to see the number of tests that has passed without the person having to read the details of each individual test. We then created a column which will store the **date** that the test has been carried out on. This will make it easier to see which version of a test is the latest if a test had been carried out multiple times. Finally, we added a column for **notes** which can be used to store additional details about the tests that does not fall into any other category. For example, if a test has failed, the notes could include a description as to why the test has failed.

Company Email System (Main Class)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Function Name | Test Aim | Prerequisite | Inputs | Expected Output(s) | Actual Output | Test Created By | Source: Spec or code inspection | Pass / Fail | Date (Completed by) | Notes |
| 301 | Main Method | Testing the user sees the correct menus when not in a project. | Set Current Project Variable to 0 | No inputs | A list of user options:   * See a list of projects * Add a new project * Viewing a project * Exit the system |  | Aidan Reed | Specification - Class Description |  |  |  |
| 302 | Main Method | Tests the user cannot see the should not be able to see the commands offered by the loop. | Create a project for the test | Select project created in prerequisite stage | The user should have access to the project and no menus. |  | Aidan Reed | Specification – Class Description |  |  |  |
| 303 | Main Method | Tests the ArrayList can grow to large size | Create 10,000 Projects | N/A | Program continues to run displaying the user command menu |  | Aidan Reed | Specification – Class Description |  |  | Tests the program does not crash or thrown an exception when handling a large number of projects. Also tests to ensure the array is not statically set. |
| 304 | List Project | Checks that the function returns the correct number of emails in the current phase. | Add 10 emails to the current project phase | Select the test project | A string with the count in the format :  New Project [Design] – 10 Emails |  | Aidan Reed | Specification – Class Description |  |  |  |
| 305 | List Project | Checks that the function can handle large email counts. | Create 100,000 emails for current project phase | Select the test project | A string with the count in the format:  New Project [Design] – 10 Emails  The execution time should be below 5 seconds. |  | Aidan Reed | Specification – Class Description |  |  | This test includes a performance test for the function to ensure the function is efficient when handling large amounts of data. |
| 306 | Add Project | Ensures the users title added as input is correctly added to the project | Set the Current Project Variable to 0 | Select the new project command and enter the Marketing Campaign title | The project should be created and display the name Marketing Campaign |  | Aidan Reed | Specification – Class Description |  |  |  |
| 307 | Add Project | Checks that the project is created if the user does not enter a name | Set the Current Project Variable to 0. | Select the new project command and do not enter title | The project should be created with no name but state project ID. |  | Aidan Reed | Specification – Class Description |  |  | The Class Description Function 3 does not state what should be displayed if the user omits to enter the name. The test ensures the Project ID is currently presented. |
| 308 | List Emails | Tests that the function returns the correct phase ID for project | Create project Orion and set phase ID TO 2 | Phase ID 1 | Returns the Display the emails for current project at phase ID 1 |  | Aidan Reed | Specification – Class Description |  |  |  |
| 309 | List Emails | Tests the function with an invalid phase ID | Create project Orion and set Phase ID to 2 | Phase ID 20000 | Should not return any emails. |  | Aidan Reed | Specification – Class Description |  |  |  |
| 310 | List Emails | Test the function with no phase ID | Create Project Orion and set Phase ID to 2 | N/A | Should return emails for the current project at phase ID 2 |  | Aidan Reed | Specification – Class Description |  |  |  |
| 311 | List Phases | Call the function from the user interface | Create Project Orion and Select Project and set phase ID.  Add 100 emails phase 1  Add 200 emails phase 2  Add 20 emails phase 3 | N/A gets current project from global variable | Returns the phases for the current project with email counts:  Add 100 emails phase 1  Add 200 emails phase 2  Add 20 emails phase 3 |  | Aidan Reed | Specification – Class Description |  |  |  |
| 312 | List Contacts | The function is called from user interface in the user context | Create Project Alpha and add 20 users | Select the project and list contacts button | Returns the list of 20 users set in the prerequisite stage |  | Aidan Reed | Specification – Class Description |  |  |  |
| 313 | List Contacts | Call function when not in project | Current project value set to 0 | N/A | The program should not crash and thrown an exception |  | Aidan Reed | Specification – Class Description |  |  | It’s important to check that the class implantation avoids error prone constructs and does not crash if the project variable is not set correctly. |
| 314 | Change Project Phase | Ensures the next phase function is called correctly | Create Project Beta and set project phase 1 | Select Next phase button | Calls the next phase function and prints out a message to say it was successful |  | Aidan Reed | Specification – Class Description |  |  |  |
| 315 | Change Project Phase | Tests to check the last phase message is displayed | Create Project November and set project phase to last phase | Select Next phase button | Calls the next phase function and prints out message informing user already in last phase |  | Aidan Reed | Specification – Class |  |  |  |
| 316 | List Phase | Test to check how the system responds to user request of viewing Phase Folders in a new project. | Test 311 completed.  Create a new project. | Select List Phase Folders | Display a list of empty folders / Display an empty list |  | Ram Raja |  |  |  |  |
| 316 | Main Method? | Test the user cannot access a nonexistent project. | None | Enter a non-existent project number. | Inability to access the project. |  | Ram Raja |  |  |  |  |
| 317 | Main Method? | Return first project | First project must exist. | Select first project. | Display first project. |  | Ram Raja | Code Inspection |  |  |  |