Software Testing Report

<Accident Data User Interface>

Lambert Niyuhire, Cody Perkins, Mitchell Reid

Table of Contents

[1.0 Unit Tests 3](#_Toc49779837)

[2.0 Coverage Report 4](#_Toc49779838)

[3.0 Requirements Acceptance Testing 5](#_Toc49779839)

# Unit Tests

| **No** | **Test Case** | **Expected Results** | **Actual Results** |
| --- | --- | --- | --- |
| **1.0** | Wordcount Functions |  |  |
| 1.1 | Test a wrong filename | Exception Handled | Exception Handled |
| 1.2 | Test empty input file | Display error message and exit | Display error message and exit |
| **2.0** | Histogram Functions |  |  |
| 2.1 | Empty input dictionary | Display error message and exit | Display error message and exit |
| 3.0 | Search and Graph Tests |  |  |
| 3.1 | Empty Graph | Either Error Message or the graph being empty |  |
| 3.2 | Search Term not applicable | Error Message |  |
| 3.2 | Minimal data for graph | Graph will make do??? |  |
| 3.3 | Graph Has Too much data | Graph will ‘zoom out’ to account for it all |  |
| 4.0 | Correct Data |  |  |
| 4.1 | Search term that IS applicable | System will return all data entries that fit Search terms |  |
| 4.2 | Standard Expected Graph | Using data selected, the graph will be made based on it |  |
| 4.3 | Too much data returned from search | Either multiple pages of returned results, or one long page that must be scrolled through. |  |
|  |  |  |  |

# Coverage Report

In our evaluation of the Accident Data User Interface, we did more than just a quick glance at the extensive unit tests we've developed. Our evaluation of coverage was thorough. We spent a great deal of time analysing how effectively our tests covered distinct statements within specific functions, the many branches in our decision-making process; the factors that determined how our programme would run.

We tested based on the expected results that most of the expected uses would be interested in. An example of this would be Testing the results of all the accidents in a certain suburb, or most common cause of accidents. And then made a graph with those results.

# Requirements Acceptance Testing

| **Software  Requirement No** | **Test** | **Implemented (Full /Partial/ None)** | **Test Results (Pass/ Fail)** | **Comments (for partial implementation or failed test results)** |
| --- | --- | --- | --- | --- |
| 1 | There is a clear hierarchy and structure to the interface | Full | Pass | Ensures alignment with design criteria |
| 2 | Wireframes and mock-ups match visual design elements | Full | Pass | A consistent visual appearance is observed |
| 3 | Data from accidents can be navigated seamlessly through the system | Full | Pass | A seamless navigation of accident data is provided by the system |
| 4 | Invalid input error handling | Full | Pass | Results are in line with expectations |
| 5 | Invalid file names or non-existent files should be displayed with appropriate messages | Full | Pass |  |
| 6 | Icons and graphics (visual elements) are displayed appropriately | Full | Pass | Fits the aesthetic of the design |
| 7 | Ensure that full path inputs are handled when handling file names | Full | Pass | In accordance with the design |
| 8 | Windows supports multi-level file names | Partial | Fail | Due to performance issues, the program fails after 100 levels |
| 9 | The Gantt chart is implemented with all activities and milestones | Full | Pass | Project features are implemented according to plan |
| 10 | Using the mock-ups as a basis for data visualization | Full | Pass | The wireframe mock-ups and diagrams in Drawio match expectations. |
| 11 | Integrated with the database according to the Gantt plan. | Full | Pass | Integrate the database seamlessly |