Software Testing Report

NSW Traffic Penalty Tool

Brianne Byer s5175100

Wonwoo Choi s5145987

Marco Querzola s5264979

Table of Contents

[1.0 Unit Tests 3](#_Toc49779837)

[2.0 Coverage Report 4](#_Toc49779838)

[3.0 Requirements Acceptance Testing 5](#_Toc49779839)

# Unit Tests

Delete the RED text and replace with your own

(In this table you fill out details about what unit tests you have done using the unittest module)

| **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 |

| **No** | **Test Case** | **Expected Results** | **Actual Results** |
| --- | --- | --- | --- |
| **1.0** | **Check textfile name function** |  |  |
| 1.1 | Test file existence | Print(“data.csv is exist”) | Print “data.csv is exist” on console |
| 1.2 | Test wrong file name | Exit run | Print “file not found or file name error” Stop run the code and exit |
| **1.2** | **Check** | Give error message | Print “file not found or file name error” Stop run the code and exit |
| 2.0 | Check start month |  |  |
| 2.1 | Test start\_month can be converted into int | **Print Start Month is correct type on console.** | **Print Start Month is correct type on console.** |
| 2.2 | Put string value(‘a’) to start\_month and run | **Print Start Month is wrong type on console.** | **Print Start Month is wrong type on console.** |
| 3.0 | Check Start year |  |  |
| 3.1 | Test start year can be converted into int | **Print Start year is correct type on console.** | **Print Start year is correct type on console.** |
| 3.2 | Put string value(‘a’) to start year and run | **Print Start year is wrong type on console.** | **Print Start year is wrong type on console.** |
| 4.0 | Check end month |  |  |
| 4.1 | Test end month can be converted into int | **Print end month is correct type on console.** | **Print end month is correct type on console.** |
| 4.2 | Put string value(‘a’) to end month and run | **Print end month is wrong type on console.** | **Print end month is wrong type on console.** |
| 5.0 | Check end year |  |  |
| 5.1 | Test end year can be converted into int | **Print end year is correct type on console.** | **Print end year is correct type on console.** |
| 5.2 | Put string value(‘a’) to end year and run | **Print end year is wrong type on console.** | **Print end year is wrong type on console.** |
| 6.0 | Check the school\_zone\_bool is Boolean type |  |  |
| 6.1 | use isinstance function check whether the school\_zone\_bool is bool type | **Print school\_zone\_bool is correct type on console.** | **Print school\_zone\_bool is correct type on console.** |
| 6.2 | Put “True” in school\_zone\_bool | **Print school\_zone\_bool is not correct type on console.** | **Print school\_zone\_bool is not correct type on console.** |
| 7.0 | Check the range\_date\_date\_format is date type by using isinstance |  |  |
| 7.1 | use isinstance function check whether the range\_date\_date\_format is bool type | **Print range\_date\_date\_format is correct type on console.** | **Print range\_date\_date\_format is correct type on console.** |
| 7.2 | Inser “2012-03-01” to range\_date\_date\_format | **range\_date\_date\_format doesn't converted into date format** | **range\_date\_date\_format doesn't converted into date format** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Coverage Report

A description of the coverage of your unit tests, including how you evaluated coverage (function, statement, branch, condition)

# Requirements Acceptance Testing

(You will need to fill out the column on the left with the requirements listed in software design documents and the columns on the right with the results of your own testing)

| **Software  Requirement No** | **Test** | **Implemented (Full /Partial/ None)** | **Test Results (Pass/ Fail)** | **Comments (for partial implementation or failed test results)** |
| --- | --- | --- | --- | --- |
| 1 | Accept multiple file names as arguments from the command line |  |  |  |
| 2 | Display the details of all valid files |  |  |  |
| 3 | Display an appropriate message if a file does not exist or if a file name is invalid |  |  |  |
| 4 | Display a message if an argument is a directory instead of a file |  |  |  |
| 5 | File name can be a simple file name or include the full path of the file with one or more levels |  |  |  |
| 6 | file names must start with an alphabetical character |  |  |  |
| 7 | Valid file name extensions must be 3 or 4 alphabetical characters preceded by a dot) |  |  |  |
| 8 | Directory/level names must start with an alphabetical character to be considered valid |  |  |  |
| 9 | The program should be able to accept as many levels for each file name as the user wants to input. This is limited only by the number of levels allowed in Windows (approximately 120) |  |  |  |