Software Testing Report

New South Wales Traffic and Penalty Analysis Tool

Daehoon Seong  
Waraphorn Srisomboonkamol  
Justin Bilao

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** | **Overall Report** |  |  |
| 1.1 | Test user-selected timeframe | Returns a Pandas DataFrame with correct timeframe. | Returns a Pandas DataFrame with correct timeframe. |
| 1.2 | Test filter for offence code | Returns a Pandas DataFrame with correct offence code filter. | Returns a Pandas DataFrame with correct offence code filter. |
| **2.0** | **Chart Distribution** |  |  |
| 2.1 | Test user-selected timeframe | Returns a Pandas DataFrame with correct timeframe. | Returns a Pandas DataFrame with correct timeframe. |
| 2.2 | Test pie-chart distribution | Returns pie-chart based on correct timeframe. | Returns pie-chart based on correct timeframe. |
| **3.0** | **Cases Captured** |  |  |
| 3.1 | Test user-selected timeframe | Returns a Pandas DataFrame with correct timeframe. | Returns a Pandas DataFrame with correct timeframe. |
| 3.2 | Test filter by offence description | Returns a Pandas DataFrame with correct offence description filter. | Returns a Pandas DataFrame with correct offence description filter. |
| 3.3 | Test camera/radar selection | Returns a Pandas DataFrame with correct selection filter. | Returns a Pandas DataFrame with correct selection filter. |
| 3.5 | Test re-entering filters | Returns a Pandas DataFrame with correct selection filter. | Returns a Pandas DataFrame with correct selection filter. |
| **4.0** | **Cases Analyse** |  |  |
| 4.1 | Test Mobile Usage Offence Over Time | Returns Line-chart with correct timeframe and filter. | Returns Line-chart with correct timeframe and filter. |
| 4.2 | Test re-selecting timeframe | Returns Line-chart with correct timeframe and filter. | Returns Line-chart with correct timeframe and filter. |
| **5.0** | **Financial Analysis** |  |  |
| 5.1 | Test user-selected timeframe | Returns a Pandas DataFrame with correct timeframe. | Returns a Pandas DataFrame with correct timeframe. |
| 5.2 | Test table of different filters | Returns a Pandas DataFrame with correct filter(Location, etc.). | Returns a Pandas DataFrame with correct filter(Location, etc.). |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Unit Tests**

Our unit test code is in test\_plotting\_functions.py. The unit tests were designed to cover the following functionalities:

1. Reading Data Files: Tests were designed to validate the functionality for reading different types of files, including valid and invalid file formats.
   1. Test Case: test\_read\_data
2. Filtering by Date Period: This feature was tested using a variety of date ranges, including missing values and wx.DateTime objects.
   1. Test Case: test\_filter\_by\_period
3. Filtering by Phone Usage in Offence Description: The tests ensure that filtering by phone-related offenses works as expected.
   1. Test Case: test\_filter\_by\_phone
4. Filtering by Offense Code: The function was tested with different types of offense codes, including numeric and alphanumeric strings.
   1. Test Case: test\_filter\_by\_off\_code
5. Filtering by Camera and Radar Types: Tests were designed to verify the functionality of filtering by different types of camera and radar devices.
   1. Test Case: test\_filter\_by\_camera\_radar
6. Grouping by Fees: This was tested for both valid and missing columns.
   1. Test Case: test\_group\_total\_fees\_by\_column\_name
7. Top 10 Offenses: Tests were designed to validate the functionality for filtering the top 10 offenses by total number.
   1. Test Case: test\_filter\_by\_top\_10
8. Filtering by Offense Description: This was tested using different kinds of strings and special characters.
   1. Test Case: test\_filter\_by\_off\_desc

Test Cases Used:  
empty\_df = **pd**.**DataFrame**()

non\_empty\_df = **pd**.**DataFrame**({

'OFFENCE\_MONTH': **pd**.**date\_range**(start='2021-01-01', periods=4, freq='D'),

'OFFENCE\_DESC': ['Speeding', 'Red Light', 'Over Limit', 'No Seatbelt'],

'TOTAL\_VALUE': [100, 200, 150, 50],

'OFFENCE\_CODE': [123123, 200, "str\_code", 999],

'TOTAL\_NUMBER': [100, 200, 300, 400]

})

date\_inputs = [

(**pd**.**Timestamp**('2022-01-01'), **pd**.**Timestamp**('2022-02-01')),

(**wx**.DateTime(2025, 1, 1), **wx**.DateTime(2025, 2, 1)),

(**pd**.**Timestamp**('2022-03-01'), **pd**.**Timestamp**('2022-04-01')),

(**pd**.**Timestamp**(''), **pd**.**Timestamp**('2022-04-01')),

(**pd**.**Timestamp**('2022-03-01'), **pd**.**Timestamp**(''))

]

tests = [

(empty\_df, True),

(non\_empty\_df, False),

]

desc\_inputs = ['', '2', '--', 'x']

fees\_test = [

(**pd**.**DataFrame**({'OFFENCE\_DESC': ['Speeding', 'Red Light', 'Over Limit', 'No Seatbelt'] }), 'OFFENCE\_DESC', "No TOTAL\_VALUE Column in Dataframe"),

(non\_empty\_df, 'OFFENCE\_MONTH', **pd**.**DataFrame**)

]

df\_phone = **pd**.**DataFrame**({

'OFFENCE\_DESC': ['mobile usage', 'Phone usage', 'speeding', 'null', None]

})

df\_code = **pd**.**DataFrame**({

'OFFENCE\_CODE': [123, '123', 124, '125', '126a']

})

filter\_by\_phone\_cases = [

(df\_phone, 2),

(**pd**.**DataFrame**({'OFFENCE\_DESC': ['speeding', 'drinking']}), 0),

(**pd**.**DataFrame**({'OFFENCE\_DESC': ['Mobile', 'PHONE']}), 2),

(**pd**.**DataFrame**({'OFFENCE\_DESC': [None, 'speeding']}), 0)

]

df\_camera\_radar = **pd**.**DataFrame**({

'OFFENCE\_DESC': ['Camera Speeding', 'Lidar Speeding', 'Radar Speeding', 'Normal Speeding'],

'CAMERA\_TYPE': [1, None, None, None],

'SPEED\_CAMERA\_IND': [None, None, 1, None]

})

file\_inputs = [("test\_data.csv", **pd**.**DataFrame**), ("file\_not\_exist.csv", "File not found or wrong file type."), ("wrong\_format.json", "File not found or wrong file type.")]

# Coverage Report

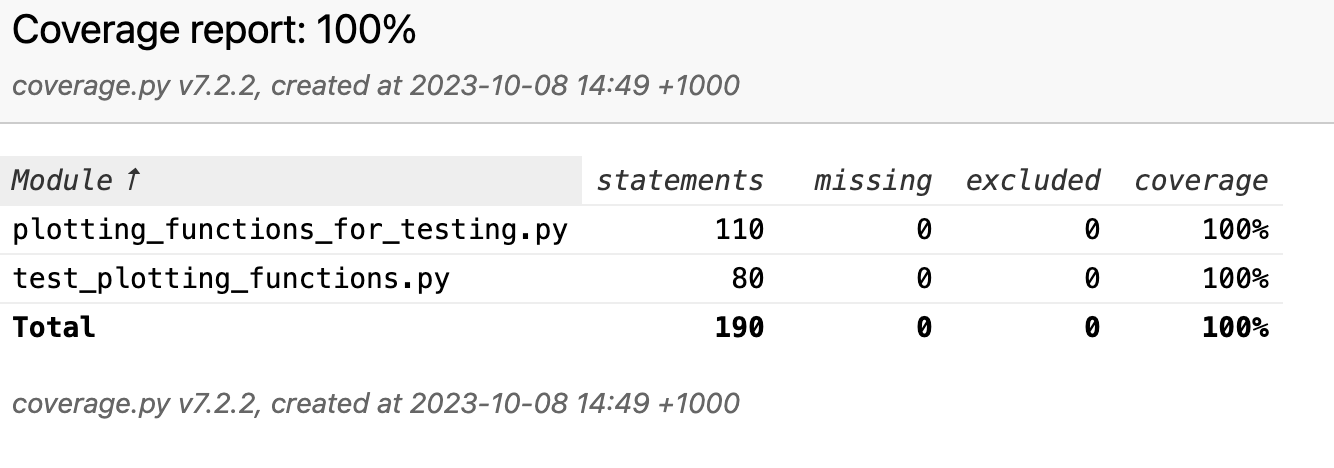
A description of the coverage of your unit tests, including how you evaluated coverage (function, statement, branch, condition)

Description of Test Coverage

**Methodology**

The test coverage for the software was evaluated using unit tests that were developed to validate the functionality of the program. Various types of test coverage were considered:

1. Function Coverage: Ensured that each function was invoked at least once.
2. Statement Coverage: Ensured that each line of code was executed at least once.
3. Branch Coverage: Ensured that both the true and false branches of if, else, try, and except statements were taken.
4. Condition Coverage: Tested the boolean expressions in decision-making constructs to evaluate both True and False outcomes.



Evaluation Metrics

The test coverage was evaluated by running these unit tests and then checking the lines of code executed, branches taken, and conditions evaluated. Our main application function for plotting and calculations were modified to cover all lines and test cases properly. Our testing parameters will have wxPython Date values and the tested functions will assert wxPython Date values. The main application functions used by our app takes in cell value and then converted to wxPython Date.

Original Function File:

def **convert\_wx\_to\_pd**(date):

from\_wx\_date = date.GetValue()

from\_date\_datetime = **datetime**.**fromtimestamp**(from\_wx\_date.GetTicks())

from\_date\_strf = from\_date\_datetime.**strftime**('%d/%m/%Y')

date = **datetime**.**strptime**(from\_date\_strf, '%d/%m/%Y')

return date

Modified Function File:

def **convert\_wx\_to\_pd**(date):

from\_date\_datetime = **datetime**.**fromtimestamp**(date.GetTicks())

from\_date\_strf = from\_date\_datetime.**strftime**('%d/%m/%Y')

date = **datetime**.**strptime**(from\_date\_strf, '%d/%m/%Y')

return date

Results

After running all the unit tests, the code coverage was found to be satisfactory, covering all lines, branches, and conditions. By carefully crafting these tests, we aimed to ensure that the software behaves as expected in a variety of scenarios.

*####Test Cases*

# Requirements Acceptance Testing

| **Software  Requirement No** | **Test** | **Implemented (Full /Partial/ None)** | **Test Results (Pass/ Fail)** | **Comments (for partial implementation or failed test results)** |
| --- | --- | --- | --- | --- |
| 1.0 | The system allows users to select a specific time-period for reporting penalty cases. | Full | Pass |  |
| 1.1 | The system provides an intuitive text-based widget for period selection. | Full | Pass |  |
| 1.2 | The system provides a table to display the list information of offences based on the offence code and period. | Full | Pass |  |
| 1.3 | The system allows exporting the tables as a PDF file. | None |  | Technical challenges were encountered during the implementation. |
| 2.0 | The system enables users to generate charts showing the distribution of cases based on offence codes for a selected period. | Full | Pass |  |
| 2.1 | The system provides an intuitive text-based widget for period selection. | Full |  |  |
| 2.2 | The systems display a pie-chart displaying the distribution of the number of the cases for each offence. | Full |  |  |
| 2.3 | The system displays the description of the offence code when hovering the mouse pointer on the offence code. | Partial |  | The requirement chart output was implemented using matplotlib which returns a whole image of a chart. |
| 2.4 | The system displays other information of the offence code when clicks on the offence code. | Partial |  | The requirement was not fully implemented due to time constraints. |
| 2.5 | The system allows exporting the charts as a PDF file. | None |  | Technical challenges were encountered during the implementation. |
| 3.0 | The system provides a filter to retrieve all cases captured by camera/radar within a user-selected timeframe. | Full | Pass |  |
| 3.1 | The system requires inputting a user-selected timeframe. | Full | Pass |  |
| 3.2 | The system includes options to select specific types of cameras or radars. | Full | Pass |  |
| 3.3 | The system displays a table based on the pre-selected filters. | Full | Pass |  |
| 3.4 | The system allows re-entering filters for display table. | Full | Pass |  |
| 3.5 | The system allows exporting the tables as a PDF file. | None |  | Technical challenges were encountered during the implementation. |
| 4.0 | Analysing the cases caused by mobile phone usage. | Full | Pass |  |
| 4.1 | The system requires inputting the type of Mobile Usage Offence to display. | Full | Pass |  |
| 4.2 | The system allows re-selecting offence types and timeframe for display. | Full | Pass |  |
| 4.3 | The system displays the line graph per selected timeframe. | Full | Pass |  |
| 4.4 | The system allows exporting the line graphs as a PDF file. | None |  | Technical challenges were encountered during the implementation. |
| 5.0 | The system offers financial analysis tools to assess the total fees collected on selected period. | Full | Pass |  |
| 5.1 | The system requires inputting a user-selected timeframe. | Full | Pass |  |
| 5.2 | The system includes an option to display a table of locations with highest total amount of penalties in descending order. | Full | Pass |  |
| 5.3 | The system allows re-selecting timeframe for display. | Full | Pass |  |
| 5.4 | The system allows exporting the tables as a PDF file. | None |  | Technical challenges were encountered during the implementation. |