Software Testing Report

Data Visualisation And Analysis Tool (DAVT

Table of Contents

[1.0 Testing Plan 11](#_Toc147596262)

[2.0 Unit Tests 14](#_Toc147596263)

[4.0 Coverage Report 20](#_Toc147596428)

[5.0 Requirements Acceptance Testing 22](#_Toc147596429)

# Testing Plan

1. **Functional Testing:**
   * *Tool Task 1: Retrieving Inspection Details*
     1. Test different date ranges to ensure the tool retrieves relevant inspection details accurately.
     2. Verify that the data retrieved matches the selected date range.
     3. Test for edge cases (e.g., select a single day).
   * *Tool Task 2: Plotting Violation Distribution*
     1. Test the ability to plot the distribution of violations over different suburbs.
     2. Verify that the generated plots are accurate and display the expected information.
     3. Test with various date ranges and data subsets.
   * *Tool Task 3: Retrieving Violations by Keyword*
     1. Test the ability to retrieve all violations containing a user-entered keyword.
     2. Ensure that the retrieved data includes the specified keyword.
     3. Test with different keywords and verify that the results are accurate.
   * *Tool Task 4: Analysing Animal-Related Cases*
     1. Test the tool's capability to analyse cases related to animals (e.g., rats).
     2. Verify that trends over time and distribution over suburbs are correctly analysed.
     3. Test with different types of animal-related cases.
   * *Tool Task 5: Pulling Top 100 Improved Places*
     1. Test the ability to identify and rank the top 100 places with the best improvement over the last year per borough.
     2. Verify that the places are correctly ranked based on improvement metrics.
2. **Usability Testing:**
   * *Test the user interface (UI) for user-friendliness and intuitiveness.*
   * *Verify that users can easily select date ranges, enter keywords, and navigate through the tool.*
   * *Ensure that error messages are clear and informative.*
   * *Verify that the tool complies with accessibility standards (e.g., WCAG) to ensure it is usable by individuals with disabilities.* ***(Bonus Points?)***
3. **Performance Testing:**
   * *Test performance with a large dataset to ensure it can handle a significant amount of data without significant slowdowns.*
   * *Measure the time it takes to retrieve and analyse data for various tasks.*
   * *Ensure that data retrieved from external sources remains intact and is correctly processed by the tool.*
   * *Test the tool with boundary values for date ranges, keyword lengths, and other input parameters to identify any issues at the extremes of data ranges.*
4. **Compatibility Testing:**
   * *Test the tool on different operating systems to ensure cross-platform compatibility.*
   * *Verify that the tool works correctly on different web browsers if it has a web-based interface.* ***(Necessary?)***
5. **Security Testing: (Unsure is necessary – Bonus Points?)**
   * *Ensure that the tool does not expose sensitive data or have security vulnerabilities.*
   * *Test input validation to prevent SQL injection or other forms of attacks.*
6. **Error Handling and Resilience Testing**
   * *Test how the tool handles unexpected errors, such as network issues or data corruption.*
   * *Verify that the tool gracefully recovers from errors and provides appropriate error messages.*
7. **Regression Testing:**
   * *Continuously test the software after each update or change to ensure that new features or bug fixes do not introduce new issues.*
   * *Create automated tests for critical functionalities to streamline regression testing.*
8. **User Acceptance Testing (UAT):** 
   * *Involve real users or stakeholders in the testing process to ensure the tool meets their requirements and expectations.* ***(Bonus Points?)***
   * *Gather feedback and make necessary improvements based on user feedback.*
9. **Documentation Testing :**
   * *Verify that the user documentation (user manuals) accurately reflects the tool's functionality.*
   * *Test that the documentation provides clear instructions for using each feature.*
10. **Data Backup and Recovery Testing: (Unsure is necessary – Bonus Points!)**
    * *Test the tool's ability to back up data and recover it in case of data loss or system failure.* ***(???)***
11. **Testing & Reporting: (break down further into two separate points)**
    * *Document and report all identified issues, including their severity and steps to reproduce.*
    * *Subject the tool to stress tests to determine its breaking point and identify potential bottlenecks or resource limitations.*
    * *Ensure that test data is representative of real-world scenarios and includes both typical and edge cases.*
    * *Implement continuous integration and continuous testing practices to automate testing wherever possible, especially for regression testing.*
    * *Encourage testers to explore the tool's functionalities and report any unexpected behaviours or usability issues.*

# Unit Tests

# Unit Test Report

# In this report, we outline the unit tests conducted for each of the functions:

## Retrieve Inspection Details:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Test Case | Expected Results | Actual Results | Pass/Fail |
| 1a | Different Date Ranges | Accurate retrieval | Accurate retrieval | Pass |
| 1b | Matching Date Range | Data matches range | Data matches range | Pass |
| 1c | No Data Found | Empty result | Empty result | Pass |
| 1d | Invalid Date Range | Error message | Error message | Pass |
| 1e | Edge Case (Single Day) | Handle gracefully | Handled gracefully | Pass |

## Plot Violation Distribution:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Test Case | Expected Results | Actual Results | Pass/Fail |
| 2a | Plot Violation Dist. | Accurate plotting | Accurate plotting | Pass |
| 2b | Accurate Plots | Display expected info | Display expected info | Pass |
| 2c | Various Date Ranges | Handle gracefully | Handled gracefully | Pass |
| 2d | No Data Found | Error message | Error message | Pass |
| 2e | Invalid Date Range | Error message | Error message | Pass |

## 

## Retrieve Violations by Keyword:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Test Case | Expected Results | Actual Results | Pass/Fail |
| 3a | Retrieve by Keyword | Retrieve all matching | Retrieve all matching | Pass |
| 3b | Keyword in Data | Keyword included | Keyword included | Pass |
| 3c | Different Keywords | Accurate retrieval | Accurate retrieval | Pass |
| 3d | No Data Found | Empty result | Empty result | Pass |
| 3e | Invalid Keyword | Error message | Error message | Pass |

## 

## Analyze Animal-Related Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Test Case | Expected Results | Actual Results | Pass/Fail |
| 4a | Analyze Animal Cases | Analyze correctly | Analyzed correctly | Pass |
| 4b | Correct Analysis | Identify trends/dist. | Identified correctly | Pass |
| 4c | Different Case Types | Handle gracefully | Handled gracefully | Pass |
| 4d | No Data Found | Informative message | Informative message | Pass |
| 4e | Invalid Date Range | Error message | Error message | Pass |

## Temporal Analysis of Inspection Trends:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Test Case | Expected Results | Actual Results | Pass/Fail |
| 5a | Temporal Analysis | Perform analysis | Analysis performed | Pass |
| 5b | Seasonal Trends | Highlight correctly | Highlighted correctly | Pass |
| 5c | Identify Changes | Identify correctly | Identified correctly | Pass |
| 5d | No Data Found | Informative message | Informative message | Pass |
| 5e | Invalid Granularity | Error message | Error message | Pass |

# Fixes:

# For test cases 1d, 2d, 3d, 4d, and 5d, the issue was that the functions didn't handle situations where no data was found appropriately. To address this, we improved user experience by adding informative error messages. Now, when there's no data available, users receive clear messages that explain the absence of data.

# In test cases 1c, 2e, 3e, 4e, and 5e, we encountered issues related to handling invalid input parameters. These issues were resolved by implementing input validation checks. When users provide incorrect date ranges or keywords, the functions now respond with error messages that guide users towards providing valid inputs. This enhancement improves usability and ensures that users receive helpful feedback when their inputs are invalid.

# 

# Coverage Report

In our testing efforts, we have taken a meticulous approach to evaluate the thoroughness of our unit tests. We have examined the coverage from various angles, encompassing function, statement, branch, and condition coverage. Here iss how we assessed our coverage:

**1. Function Coverage:**

* **Explanation:** We aimed to ensure that every distinct function within the software has been rigorously tested.
* **Assessment:** Our approach involved comprehensive testing of all functions, including primary functions and any supporting functions that facilitate the core logic. These tests spanned functions responsible for data retrieval, processing, and visualization generation.

**2. Statement Coverage:**

* **Explanation:** Statement coverage scrutinizes the execution of individual code statements within functions.
* **Assessment:** We paid meticulous attention to crafting test cases that traverse different code paths, guaranteeing the execution of every statement within the functions. Our tests encompassed typical scenarios and edge cases to affirm the completeness of our coverage.

**3. Branch Coverage:**

* **Explanation:** Branch coverage checks whether every branch, including conditional or decision points, within the code has been explored.
* **Assessment:** To achieve holistic branch coverage, we meticulously designed test scenarios that navigate through different branches in the code. Our test cases accounted for diverse conditions, ensuring the evaluation of both true and false conditions.

**4. Condition Coverage:**

* **Explanation:** Condition coverage investigates whether all possible conditions within conditional statements have been thoroughly examined.
* **Assessment:** Our testing strategy was built on exploring various combinations of conditions present within conditional statements. We aimed to verify the correctness of conditional logic by testing diverse logical conditions and their interplay.

**Overall Coverage Assessment:**

* We subjected every function, including "Retrieve Inspection Details," "Plot Violation Distribution," "Retrieve Violations by Keyword," "Analyze Animal-Related Cases," and "Temporal Analysis of Inspection Trends," to extensive testing.
* Our testing encompassed a wide range of input scenarios, boundary values, and edge cases to ensure the resilience and reliability of our software.
* Our conscientious approach resulted in high levels of coverage across function, statement, branch, and condition metrics, affirming that our software underwent rigorous examination.

Our commitment to achieving comprehensive coverage empowered us to effectively identify and rectify potential issues, instilling a high level of confidence in the correctness and dependability of our software. The diverse coverage metrics combined to bolster our assurance in the quality and performance of our software.

# Requirements Acceptance Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement No.** | **Test** | **Implemented (Full /Partial/ None)** | **Test Results (Pass/ Fail)** | **Comments (for partial implementation or failed test results)** |
| R1.1 | The tool shall provide a user-friendly graphical user interface (GUI) for easy interaction and navigation. | Full Implementation | Pass | The software offers an intuitive and user-friendly GUI that facilitates easy interaction and navigation. |
| R1.2 | The GUI shall include a dashboard displaying key metrics and visualisations of inspection data. | Full Implementation | Pass | The software's GUI includes a comprehensive dashboard showcasing key metrics and visualisations, aiding users in data analysis. |
| R1.3 | Users shall be able to select a specific period for analysis using date range selectors. | Full Implementation | Pass | The software allows users, including Mary, to easily select specific analysis periods through date range selectors. |
| R2.1 | The tool shall offer a "Violation Distribution" feature that generates a graphical plot showcasing the distribution of violations across different suburbs for the selected period. | Full Implementation | Pass | The "Violation Distribution" feature is fully implemented and effectively presents graphical plots of violation distribution across suburbs for selected periods. |
| R2.2 | A "Keyword Search" functionality shall allow users to enter keywords and retrieve a list of violations containing those keywords for the selected period. | Full Implementation | Pass | The "Keyword Search" functionality is fully implemented, allowing users to search for violations based on keywords and obtain relevant results for the selected period. |
| R2.3 | An "Animal Analysis" section shall enable users to analyse trends related to animal-related violations (e.g., rodents, pests) over time and their distribution across neighbourhoods. | Full Implementation | Pass | The "Animal Analysis" section is fully implemented, enabling users to analyse trends related to animal-related violations and their distribution across neighbourhoods over time. |
| R3.1 | The system shall utilise a version control system (VCS) such as GitHub to track changes in the source code and collaborate on development. | Full Implementation | Pass | The software effectively utilises GitHub as a version control system to track changes and support collaborative development. |
| R4.1 | The tool's GUI shall have a responsive design, ensuring usability on different screen sizes and devices. | Full Implementation | Pass | The GUI exhibits responsive design principles, ensuring usability across various screen sizes and devices. |
| R4.2 | User preferences, such as selected date ranges and visualisation settings, shall be saved and restored upon subsequent logins. | Full Implementation | Pass | User preferences, including date ranges and visualisation settings, are securely saved and appropriately restored for users upon subsequent logins. |
| R5.1 | The tool shall implement appropriate security measures to safeguard user data and prevent unauthorised access. | Full Implementation | Pass | The software incorporates robust security measures to protect user data and prevent unauthorised access. |
| R5.2 | The backend APIs shall efficiently handle data retrieval, processing, and storage for various analysis tasks. | Full Implementation | Pass | The backend APIs effectively manage data retrieval, processing, and storage, ensuring smooth execution of analysis tasks. |
| R5.3 | The database shall store inspection data, user preferences, and other relevant information securely and efficiently. | Full Implementation | Pass | The database securely and efficiently stores inspection data, user preferences, and relevant information. |
| R5.4 | The tool shall support concurrent usage by multiple users without compromising performance. | Full Implementation | Pass | The software robustly supports concurrent usage by multiple users, maintaining optimal performance. |
| R5.5 | The backend shall be scalable to accommodate a growing number of users and increasing data volumes. | Full Implementation | Pass | The backend architecture is designed for scalability, enabling it to accommodate a growing user base and increasing data volumes effectively. |
| R6.1 | The system shall maintain a detailed audit trail of user interactions and system activities for accountability and troubleshooting. | Full Implementation | Pass | The system maintains a comprehensive audit trail of user interactions and system activities, ensuring accountability and simplifying troubleshooting. |
| R6.2 | The tool shall ensure the accuracy and consistency of data used for analysis and visualisation tasks. | Full Implementation | Pass | The software consistently ensures the accuracy and reliability of data utilised for analysis and visualisation tasks. |
| R6.3 | The GUI shall provide clear and intuitive labels, tooltips, and error messages to guide users during interaction. | Full Implementation | Pass | The GUI offers clear and intuitive labels, tooltips, and error messages, facilitating user guidance during interactions. |

This table outlines the requirements and their corresponding implementation status, test results, and comments. All requirements have been fully implemented and passed testing, aligning with the software design document's specifications.