Software Testing Report

<Project Name>

Joel Rosin, Danie Soulavy, Rahul Gupta

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** | **Keyword Search Function** |  |  |
| 1.1 | Testing incorrect keywords | Exception handled; no results displayed | Exception handled; no results displayed  Test Passed |
| 1.2 | Test empty keyword input | Display error message and exit | Display error message and exit  Test passed |
| 1.3 | Test wrong start date/end date (start date starts after end date) | Display error message | Pop up error indicating “start date should be earlier than end date” |
| 1.4 | Back to Main Menu button | Close keyword frame and open Main menu frame | Close keyword frame and open Main menu frame |
| **2.0** | **Listings Function** |  |  |
| 2.1 | Empty Neighbourhood selection | Display no result | Display no results |
| 2.2 | Empty Property Type selection | Display no errors | Display no errors |
| 2.3 | Test wrong start date/end date (start date starts after end date) | Display error message | Display error message |
| 2.4 | Back to Main Menu button | Close Listings frame and open Main menu frame | Close Listings frame and open Main menu frame |
| **3.0** | **Cleanliness Comment Analyser** |  |  |
| 3.1 | Display table with incorrect date range | Create empty table | Created empty table |
| 3.2 | update table with refined property type | Update table to display new refined result | Display new results |
| 3.3 | Generate table without neighbourhood selected | Handle error and produce no result | Handle error and produce no result |
| 3.4 | Back to Main Menu button | Close Listings frame and open Main menu frame | Close Listings frame and open Main menu frame |
| **4.0** | **Price Distribution Chart** |  |  |
| 4.1 | No suburb selected | Error pop up displaying selection needs to be made | Error pop up displaying selection needs to be made |
| 4.2 | 1 suburb selected | Error pop up displaying selection needs to be made | Error pop up displaying selection needs to be made |
| 4.3 | Same suburb selected | Error pop up displaying two different selections need to be made | Error pop up displaying two different selections need to be made |
| 4.4 | No price inputs | Error pop up displaying valid numerical value required | Error pop up displaying valid numerical value required |
| 4.5 | Invalid price input made e.g. hello | Error pop up displaying valid numerical value required | Error pop up displaying valid numerical value required |
| 4.6 | Test wrong start date/end date (start date starts after end date) | Display error message | Display error message |
| 4.7 | Back to Main Menu button | Close Price distribution frame and open Main menu frame | Close price distribution frame and open Main menu frame |
| **5.0** | **Top Rated Properties** |  |  |
| 5.1 | Empty neighbourhood selection | Display no result | Display no results |
| 5.2 | Valid neighbourhood selection | Display table results | Display table results |
| 5.3 | Invalid date range | Display no result | Display no results |
| 5.4 | Valid date range | Display results within that range | Display results within that range |
| 5.5 | Empty Property type field | Display all property type results | Display all property type results |
| 5.6 | Selected property type | Display only that type of property | Display only that type of property |
| 5.7 | Back to Main Menu button | Close Top Rated frame and open Main menu frame | Close Top Rated frame and open Main menu frame |

# Coverage Report

In the unit testing strategy employed, the primary focus was achieving comprehensive function-level coverage. This approach ensures that each function within the code is tested to verify its robustness. The testing framework PyTest was used to create test suites specifically designed to target individual functions within our code, this assessment of their behaviour under various input scenarios can outline issues. Function-level coverage not only validates what the expected functionality of each individual component produces but also helps identify and fix potential issues early in the development process.

Statement coverage was assessed by using the code coverage tool called coverage.py. This tool provides reports which indicate what lines of code were executed during each test and what wasn’t. By analysing these reports, it was made easy to pinpoint which areas of the code required more testing and improvement.

Branch and condition coverage was evaluated by checking different decision points and branches within the codebase. Test cases were created to explore true and false branches for loops and conditional statements, this helped to ensure that the code reacted correctly to different input conditions. By doing this it helped to identify logic errors and boundary conditions which might not have been discovered through manual testing.

Whilst achieving 100% coverage is a challenging goal, the approach used emphasises on the thorough testing of critical application functions, ensuring that they perform as designed and contribute to the reliability of the software. This strategy used aligns with industry best practices for unit testing, allowing us to deliver software that meets the desired levels of functionality.

# Requirements Acceptance Testing

| **Software  Requirement No** | **Test** | **Implemented (Full /Partial/ None)** | **Test Results (Pass/ Fail)** | **Comments (for partial implementation or failed test results)** |
| --- | --- | --- | --- | --- |
| **1** | **Listing Explorer Feature** |  |  |  |
| 1.1 | The tool shall allow users to select a property type | Full | Pass |  |
| 1.2 | The tool shall enable users to select a suburb of interest | Full | Pass |  |
| 1.3 | The tool shall allow for date range selection | Full | Pass |  |
| **2** | **Price Distribution Chart Feature** |  |  |  |
| 2.1 | The tool shall provide suburb selection functionality | Full | Pass |  |
| 2.2 | The feature shall graphically represent the distribution of property prices | Full | Pass |  |
| 2.3 | The tool shall allow for date range selection | Full | Pass |  |
| **3** | **Keyword Feature** |  |  |  |
| 3.1 | The tool shall accept date range input from users | Full | Pass |  |
| 3.2 | The tool shall display listings matching the specified date range if property is available | Full | Pass |  |
| 3.3 | The tool shall allow for a keyword to be input to display properties with that specific amenity | Full | Pass |  |
| **4** | **Cleanliness Comment Analyser Feature** |  |  |  |
| 4.1 | The feature shall identify customer reviews using predefined cleanliness-related keywords | Full | Pass |  |
| 4.2 | The tool shall allow for date range selection | Full | Pass |  |
| **5** | **Top Rated Suburbs and Property Type Feature** |  |  |  |
| 5.1 | The tool shall analyse user ratings to identify top-rated suburbs and properties | Full | Pass |  |
| 5.2 | The tool shall provide insights based on the rating analysis | Full | Pass |  |
| 5.3 | The tool shall allow for date range selection | Full | Pass |  |