Software Testing Report

Sydney’s Airbnb Data App

**Vishesh Sagar s5247450**

**Hayden Baker S5278212**

**Zachary Hardman S5134726**

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** | **Report all Listings Function** |  |  |
| **1.1** | **t\_convert\_price\_to\_number** Verifying that the function converts a price string to an integer correctly. | The function should correctly convert a price string (e.g., "$100") to an integer (e.g., 100). | The function successfully converts the price string to an integer. |
| **1.2** | **t\_calculate\_average\_price** Checking that the function calculates the average price from a series of price strings. | The function should correctly calculate the average price from a series of price strings. | The function successfully calculates the average price. |
| **1.3** | **t\_filter\_listings\_by\_suburb** Ensuring that the function filters the listings data frame by the provided suburb name. | The function should return a DataFrame containing only rows with the specified suburb name in the 'suburb' column. | The function successfully filters listings by the provided suburb name. |
| **1.4** | **t\_get\_suburb\_listing** Testing if the function returns the correct average price and count for a single suburb. | The function should return the correct average price and count for a single suburb based on the data provided. | The function successfully returns the correct average price and count for a single suburb |
| **1.5** | **t\_get\_all\_suburbs\_average\_price** Checking that the function returns correct average prices and counts for all provided suburbs. | The function should return correct average prices and counts for all provided suburbs based on the data provided. | The function successfully returns correct average prices and counts for all provided suburbs. |
| **2.0** | **Prices Display Function** |  |  |
| **2.1** | Ensure the function **date\_filtered\_listings** successfully filters listings within a specific date range, returning only entries within the given dates. | The function should return a DataFrame containing listings with dates between start\_date and end\_date. | The function successfully filters listings by the specified date range. |
| **2.2** | **t\_handle\_empty\_dataframe**  Verify that the function **date\_filtered\_listings** gracefully handles an empty DataFrame, returning another empty DataFrame without errors. | The function should handle an empty DataFrame as input without raising errors and return an empty DataFrame as output. | The function gracefully handles an empty DataFrame and returns another empty DataFrame. |
| **2.3** | **t\_different\_date\_format**  Confirm that the function **date\_filtered\_listings** can accurately interpret and handle different date formats, filtering listings appropriately based on the provided date range. | The function should correctly interpret and filter listings based on different date formats in the provided date range. | The function accurately interprets and handles different date formats, filtering listings as expected. |
| **3.0** | **Search Keywords Function** |  |  |
| **3.1** | **t\_keyword\_filtered\_listings** Ensure the function successfully filters listings containing a specific keyword within selected attributes. | The function should return a DataFrame containing only rows where the specified keyword is found in the selected attributes. | The function successfully filters listings based on the keyword and selected attributes. |
| **3.2** | **t\_date\_filtered\_listings** Verify that the function accurately filters listings based on a specified date range. | The function should return a DataFrame containing only rows with dates falling within the specified range. | The function successfully filters listings based on the date range. |
| **3.3** | **t\_find\_keyword\_listings** Confirm the function proficiently merges filtered datasets, adhering to both keyword and date criteria. | The function should return a merged DataFrame that satisfies both the keyword and date criteria. | The function successfully merges filtered datasets based on keyword and date criteria. |
| **4.0** | **Cleanliness Function** |  |  |
| **4.1** | **keyword\_filtered\_reviews** Check if the function filters reviews with positive and negative keywords correctly. | The function should return a DataFrame containing only rows with comments that include positive or negative keywords. | The function successfully filters the reviews based on positive and negative keywords. |
| **4.2** | **Count\_keywords\_in\_reviews** Validate if the function correctly counts the occurrences of positive and negative keywords and adds the counts as new columns in the dataframe. | The function should add 'pos\_keyword\_count' and 'neg\_keyword\_count' columns to the DataFrame with counts of positive and negative keywords. | The function correctly counts and adds the keyword counts to the DataFrame. |
| **4.3** | **find\_by\_cleanliness** Confirm that the function groups by 'listing\_id', sums keyword counts, and calculates the percentage of positive keywords correctly. | The function should group reviews by 'listing\_id', sum the keyword counts, and calculate the percentage of positive keywords. | The function successfully groups, sums, and calculates the percentages of positive keywords. |
| **5.0** | **Search for Reviews** |  |  |
| **5.1** | **reviews\_by\_name**  Verify that the function filters reviews by reviewer name correctly. | The function should return a DataFrame containing reviews from the specified reviewer name.  Sample DataFrame with reviewer names 'Alice', 'Bob', 'Alice', 'Charlie'.  Filter reviews by 'Alice'.<br>2. Verify the shape and content of the resulting DataFrame.<br>3. Filter reviews by 'NoName'.<br>4. Verify the shape of the resulting DataFrame. | - For 'Alice', the function returns a DataFrame with a shape of (2, 2) containing reviews from 'Alice'.<br>- For 'NoName', the function returns an empty DataFrame with a shape of (0, 2). |

# Coverage Report

1. **Report all Listings Function**

**Upload here Screenshot of functions**

**There are 5 functions are executed during testing:**

**1. t\_convert\_price\_to\_number - PASSED**

**Verified that the function correctly converts a price string into an integer.**

**2. t\_calculate\_average\_price - PASSED**

**Checked that the function accurately calculates the average price from a series of price strings.**

**3. t\_filter\_listings\_by\_suburb - PASSED**

**Ensured that the function effectively filters the listings dataframe based on the provided suburb name.**

**4. t\_get\_suburb\_listing – PASSED**

**Tested whether the function correctly returns the average price and count for a single suburb.**

**5. t\_get\_all\_suburbs\_average\_price - PASSED**

Confirmed that the function accurately returns the correct average prices and counts for all the provided suburbs.

1. **Prices Display Function**

**Upload here Screenshot of functions**

**There are 3 functions are executed during testing:**

**1. t\_date\_range\_filter - PASSED**

**Confirm that the function 'date\_filtered\_listings' effectively filters listings within a specified date range, returning only entries falling within the specified dates.**

**2. t\_handle\_empty\_dataframe - PASSED**

**Verify that the function 'date\_filtered\_listings' gracefully manages an empty DataFrame, returning another empty DataFrame without encountering errors.**

**3. t\_different\_date\_format - PASSED**

**Ensure that the function 'date\_filtered\_listings' can accurately interpret and handle various date formats, correctly filtering listings based on the provided date range.**

1. **Search Keywords Function**

**Upload here Screenshot of functions**

**There are 3 functions are executed during testing:**

1. **- t\_keyword\_filtered\_listings - PASSED**

**Confirm that the function effectively filters listings that contain a particular keyword within the chosen attributes.**

1. **t\_date\_filtered\_listings - PASSED**

**Validate that the function accurately filters listings based on a specified date range.**

1. **t\_find\_keyword\_listings - PASSED**

**Ensure that the function adeptly merges filtered datasets while adhering to both keyword and date criteria.**

1. **Cleanliness Function**

**Upload here Screenshot of functions**

**There are 3 functions are executed during testing:**

**1. The 'keyword\_filtered\_reviews(reviews\_df)' function passed successfully. It verifies the accurate filtration of reviews based on positive and negative keywords.**

**2. The 'count\_keywords\_in\_reviews(reviews\_df)' function also passed successfully. It validates the accurate counting of occurrences of positive and negative keywords and the addition of these counts as new columns in the dataframe.**

**3. The 'find\_by\_cleanliness(reviews\_dataframe)' function has successfully passed. It confirms the correct grouping by 'listing\_id,' summing keyword counts, and calculating the percentage of positive keywords.**

1. **Search for Reviews**

**Upload here Screenshot of functions**

**There is one function executed during testing:**

**1. reviews\_by\_name - The function accurately filters reviews based on the reviewer's name.**

**We assessed the effectiveness of our tests by using various evaluation methods. These methods included:**

**1. Function Coverage: This involved making sure that we tested every function in our code.**

**2. Statement Coverage: We ensured that every statement within our code was executed at least once during testing.**

**3. Branch Coverage: Our testing also involved verifying that every branch in the code, such as if-else statements, was executed.**

**4. Condition Coverage: We made certain that each logical condition was evaluated at least once for both true and false scenarios.**

**By employing these methods, we were able to guarantee comprehensive coverage of the main functionalities and scenarios in our unit tests. This, in turn, assured the quality and stability of our software.**

# 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) |  |  |  |