Software Testing Report

<Project Name>

Student Names

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** | **Data loading Functions** |  |  |
| 1.1 | Loading a wrong database | Display error message and exit | Display error message and exit |
| 1.2 | Test without existing SQL database | Exception handled: pass and print "Not Exist!" | Exception handled: pass and print "Not Exist!" |
| 1.3 | Test with already inserted SQL database | Exception handled: pass and print "Data Already Exists!" | Exception handled: pass and print "Data Already Exists!" |
| **2.0** | **Search and filter Functions** |  |  |
| 2.1 | User tries to input past date | System won't allow | System won't allow |
| 2.2 | Test if user can input a price range | Display all properties within that price range | Display all properties within that price range |
| 2.3 | Test a specific cleanliness range | Display all properties corresponding to that cleanliness range | Display all properties corresponding to that cleanliness range |
| 2.4 | Test a non-existing cleanliness range | Display nothing | Display nothing |
| 2.5 | Test a specific keyword | Display all properties containing that keyword | Display all properties containing that keyword |
| 2.6 | Test an unknown keyword | Display nothing | Display nothing |
| 2.7 | Test if user can input a score rating range | Display properties with the score rating input and above | Display properties with the score rating input and above |
| 2.9 | Test if users can look for properties’ reviews | All reviews are displayed in properties information | All reviews are displayed in properties information |
| **3.0** | **Booking Functions** |  |  |
| 3.1 | Test if user can book a property | Booking window pop up with confirmation | Booking window pop up with confirmation |
| 3.2 | Test when user book a property already booked | System won’t allow and display “already booked” | System won’t allow and display “already booked” |
| 3.3 | Test when user cancels booking | Confirmation windows pop up | Confirmation windows pop up |
| **4.0** | **Communication Functions** |  |  |
| 4.1 | Test when user want to communicate with owner | Text bubble shows up | Text bubble shows up |
| 4.2 | Test if users can turn on notification | Sound notification whenever user receive a message from owner | Sound notification whenever user receive a message from owner |
| **5.0** | **Performance Functions** |  |  |
| 5.1 | Test response time between pages | Respond time to switch pages is 2 to 3 seconds | Respond time to switch pages is 2 to 3 seconds |
| 5.2 | Test respond time loading dataset | Dataset should load up to 10 seconds | Dataset should load up to 10 seconds |
| 5.3 | Test respond time display data | Data should display in 2 seconds | Data should display in 2 seconds |
| **6.0** | **Security Functions** |  |  |
| 6.1 | Check for vulnerabilities is dataset and databases | No vulnerabilities show up | No vulnerabilities show up |
| 6.2 | Test if users can have access to only their personal data | System won't allow user to access another user’s data information | System won't allow user to access another user’s data information |

Feature 1: Report the information of all listings in a specific suburb

import pytest  
from unittest.mock import MagicMock, Mock  
import sqlite3  
import wx  
  
from f1 import Retrieve\_By\_Time  
  
  
class MockGUI:  
 def GetValue(self):  
 return "2022-01-01"  
  
  
@pytest.fixture  
def setup\_retrieve\_by\_time(monkeypatch):  
 con = MagicMock(spec=sqlite3.Connection)  
 cur = MagicMock(spec=sqlite3.Cursor)  
  
 con.cursor.return\_value = cur  
 cur.fetchall.return\_value = []  
  
 monkeypatch.setattr("sqlite3.connect", lambda \_: con)  
  
 # Mock wx.App and instantiate it  
 class MockApp(wx.App):  
 def OnInit(self):  
 return True  
  
 mock\_app = MockApp(redirect=False)  
 monkeypatch.setattr(wx, 'App', MockApp)  
  
 # Mock wx.MessageBox  
 monkeypatch.setattr(wx, 'MessageBox', Mock())  
  
 frame = Retrieve\_By\_Time(None)  
 frame.file1 = MockGUI()  
 frame.file2 = MockGUI()  
  
 return frame, con, cur  
  
  
def test\_search(setup\_retrieve\_by\_time):  
 frame, \_, cur = setup\_retrieve\_by\_time  
  
 # Mocking the GetValue methods  
 frame.file1.GetValue = Mock(return\_value='2022-01-01')  
 frame.file2.GetValue = Mock(return\_value='2023-01-01')  
  
 frame.search(Mock())  
  
 expected\_sql = ("select \* from Listings\_Dec18 where id in "  
 "(select distinct listing\_id from Calendar\_Dec18 where "  
 "substr(date,1,4) || substr(date,6,2) || substr(date,9,2) "  
 "between '20220101' and '20230101')")  
  
 cur.execute.assert\_called\_with(expected\_sql)

A screen shot of a computer

Description automatically generated

Feature 4: Analysing how many customers commented on factors related to cleanliness

import pandas as pd  
import pytest  
from f4 import Cleanliness\_Analysing  
from unittest.mock import create\_autospec  
  
  
def mock\_read\_csv(\*args, \*\*kwargs):  
 data = {  
 'id': [1, 2, 3],  
 'name': ['a', 'b', 'c'],  
 'property\_type': ['Apartment', 'House', 'Condo'],  
 'listing\_id': [1, 2, 3],  
 'reviewer\_name': ['Alice', 'Bob', 'Charlie'],  
 'comments': ['Very clean!', 'Somewhat clean', 'Not clean at all'],  
 }  
 return pd.DataFrame(data)  
  
@pytest.fixture  
def frame(mocker):  
 mocker.patch("f4.Cleanliness\_Analysing.setup\_ui") # Mock setup\_ui  
 instance = create\_autospec(Cleanliness\_Analysing, instance=True) # Create a mock instance of the class  
 mocker.patch.object(instance, 'figure', create=True) # Mock figure attribute after instance creation  
 return instance  
  
  
@pytest.fixture  
def mock\_dataframe(mocker):  
 mocker.patch('pandas.read\_csv', side\_effect=mock\_read\_csv) # Mock read\_csv method  
  
def test\_update\_chart(frame, mock\_dataframe):  
 selected\_keyword = "clean"  
 # Test will fail if any exception occurs in the method  
 frame.update\_chart(selected\_keyword)  
  
@pytest.fixture  
def mock\_draw\_chart\_data():  
 data = {  
 'property\_type': ['Apartment', 'House'],  
 'number\_of\_people': [5, 10]  
 }  
 return pd.DataFrame(data)  
  
def test\_draw\_chart(frame, mock\_draw\_chart\_data):  
 selected\_keyword = "clean"  
 # Test will fail if any exception occurs in the method  
 frame.draw\_chart(mock\_draw\_chart\_data, selected\_keyword)

A screen shot of a computer

Description automatically generated

Feature 5: Show the rating of properties based on the customers’ experience and their satisfaction toward the property

import wx  
import pytest  
from f5 import Properties\_Rating  
  
@pytest.fixture(scope='function')  
def app():  
 app = wx.App(False)  
 yield app  
 # Cleanup  
 app.Destroy()  
  
  
@pytest.fixture(scope='function')  
def frame(app):  
 frame = Properties\_Rating(None)  
 yield frame  
 # Cleanup  
 frame.Destroy()  
  
  
def test\_initialization(frame):  
 # Ensure that the frame is shown  
 frame.Show()  
 wx.Yield()  
 assert frame.IsShown()  
  
  
def test\_setup\_ui(frame):  
 # Test the setup\_ui function  
 frame.setup\_ui()  
 assert frame.grid.GetNumberCols() == 5  
  
  
def test\_get\_page(frame):  
 # Test the get\_page function  
 frame.page = 0  
 frame.index = [str(i) for i in range(1, 21)]  
 frame.get\_page(None, 1)  
 assert frame.page == 1  
 assert frame.index == [str(i) for i in range(21, 41)]  
  
  
def test\_main\_button(frame):  
 # Redefining the Signal class to accommodate 'emitted' attribute  
 class Signal:  
 def \_\_init\_\_(self, name=None):  
 self.name = name  
 self.emitted = False  
  
 def connect(self, func):  
 self.slot = func  
  
 def emit(self):  
 if hasattr(self, "slot"):  
 self.slot()  
 self.emitted = True  
  
 # Initialize the mock signal  
 frame.return\_signal = Signal(name='Return Back')  
  
 def mock\_signal\_handler():  
 frame.return\_signal.emitted = True  
  
 # Connect the mock handler to the signal  
 frame.return\_signal.connect(mock\_signal\_handler)  
  
 # Emitting button event  
 frame.mainButton.ProcessEvent(wx.CommandEvent(wx.EVT\_BUTTON.typeId, frame.mainButton.GetId()))  
 wx.Yield()  
  
 # Assertion  
 assert frame.return\_signal.emitted

A screen shot of a computer

Description automatically generated

# 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 | The app shall allow users to input their travel dates and the desired suburb | Full | Pass |  |
| 2 | The app shall display all the available properties listed corresponding to the users’ input | Full | Pass |  |
| 3 | Each property shall display their price and users shall be able to sort properties by their prices | Full | Pass |  |
| 4 | The app shall provide a feature where users can enter a certain keyword, such as pool or garden, and the app shall display all properties listed containing these keywords. | Full | Pass |  |
| 5 | The app shall be able to perform an analysis based on the comments and reviews the customers left on properties. For example, the app shall be able to display all comments about the cleanliness of a property or the satisfaction of customers regarding the property and calculate the average cleanliness or satisfaction with stars. | Full | Pass |  |
| 6 | The app shall allow communication between the owner and the guests (users) | Partial | Pass | Sometime the owner may not reply. |