CSE 565 Assignment 5

Dr.Lynn Robert Carter

Kirity Ganga (1222577863)
Last Date Submission: 9th April, 2023

Task 1

Uncompress Applications.zip file to extract all the directories present in the file which include:

- AppV1: It is the directory of the original application file. Go into the "AppV1" directory and double-click on the "AppV1.exe" file to open the application. This application has a main page that provides basic information about Adobe Acrobat and two buttons to navigate to different pages of the application: Adobe Photoshop (Second Page) and Adobe Illustrator (Third Page).
  The Second Page, or the Adobe Photoshop page, has two radio buttons to select image formats (PNG or JPG), a "Set" button to apply the selected configuration, and a "Clear" button to reset the selection. There is also a button to navigate back to the main page.
  The Third Page, or the Adobe Illustrator page, has a slider that rotates an arrow image above it. There is also a button to navigate back to the main page.
  In summary, the "AppV1" directory contains a simple application that allows users to navigate between three pages and perform various actions, such as selecting image formats and rotating an image.

- AppV2: It is a version of AppV1 with some minor changes in its features. Go into the "AppV1" directory and double-click on the "AppV1.exe" file to open the application. Some ways AppV2 is different from AppV1 are:
  - Unlike AppV1 you cannot navigate to page 3 from the main page and can only navigate to page 2.
  - Unlike AppV1 you cannot navigate to the main page from page 2 and can only navigate to page 3.
  - Unlike AppV1 the slider in page 3is not horizontal and is vertical.
  - The AppV1 window is smaller than the AppV2 window.

  Some ways AppV2 and AppV1 are similar:
  - Like AppV1 both have the same messages in the text boxes present in page 1 and page 2.
  - Like AppV1 you have 2 options or radio buttons to select file download format.
  - Like AppV1 all share the same images present across applications.

Both the applications are created and developed using the Tkinter web framework library in python and then converted into executable files.

Uncompress TestApp.zip file to extract testcases python file:

- TestApp: "TestApp" directory contains two python files named test_cases_gui.py and test_cases_non_gui.py and a directory of test_images that contains images for validation. For this assignment we are only GUI testing and thus will not care about test_cases_non_gui.py file.

Go into the TestApp directory and run the python file test_cases_gui.py with the path of the application that is to be tested sent as an argument. For example:

```
~ python test_cases_gui.py <complete absolute path of the application>
```

Task 2

For this assignment, **PyAutoGUI** was used to record test cases and report the results. PyAutoGUI is a Python library for automating tasks on your computer. It can be used to automate mouse clicks, keyboard strokes, and other GUI interactions. PyAutoGUI has several functions that can be used to control the mouse and keyboard, such as moveTo(), click(), write(), and press().
These functions can be used to simulate user actions, such as clicking a button, typing text into a field, or pressing a key. One of the key features of PyAutoGUI is its ability to take screenshots and recognize images within those screenshots. This makes it possible to automate tasks that involve looking for specific images or patterns on the screen.
PyAutoGUI can also be used to control other applications running on your computer, such as web browsers, video players, and games. PyAutoGUI is a cross-platform library and works on Windows, macOS, and Linux. It is easy to install using pip, and comes with detailed documentation and examples to help you get started. However, it's important to note that PyAutoGUI is a powerful tool that can be used for both good and bad purposes, so use it responsibly and ethically.

Installation and Execution

To install PyAutoGUI, on command prompt pip install the library:

```
~ pip install PyAutoGUI
```

PyAutoGUI library is dependent on Opencv library and thus ensure user has opencv library installed in the system:

```
~ pip install pillow
~ pip install opencv-python
```

We also installed pyperclip library to execute clipboard copy/paste actions in file:

```
~ pip install pyperclip
```

Go into the TestApp directory and run the python file test_cases_gui.py with the path of the application that is to be tested sent as an argument. For example:

```
~ python test_cases_gui.py <complete absolute path of the application>
```

Test Cases:

A total of 12 test cases were developed to automate GUI testing:

1. Check if the application is executable: This test case checks whether the application can be executed or not.
2. Check if the Main Page (First Page) of Application Opens: This test case verifies whether the main page of the application is opening or not.
3. Check if the Second page is visited from the First Page: This test case checks whether the user is able to navigate from the main page to the second page of the application through the button.
4. Check if the Main page is visited from Second Page: This test case checks whether the user is able to navigate back to the main page from the second page of the application through the button.
5. Check if the Third page is visited from the Main Page: This test case checks whether the user is able to navigate from the main page to the third page of the application through the button.
6. Check if the Main page is visited from Third Page: This test case checks whether the user is able to navigate back to the main page from the third page of the application through the button.
7. Verify Text Present on Main Page: This test case checks whether the main page of the application contains the expected text or not.
8. Verify RadioButtons to download images: This test case verifies whether the radio buttons for downloading images are working or not.
9. Verify Clear Button to reset configuration: This test case verifies whether the clear button is resetting the configuration of the application or not.
10. Verify Slider that rotates image anti-clockwise: This test case verifies whether the slider for rotating the image anti-clockwise is working or not.
11. Verify Checkbox to rotate image clockwise: This test case verifies whether the checkbox for rotating the image clockwise is working or not.
12. Close the Application: This test case checks whether the application can be closed or not.

Results

AppV1 Results:

Command: `~ python test_cases_gui.py <complete absolute path of the application>/AppV1`

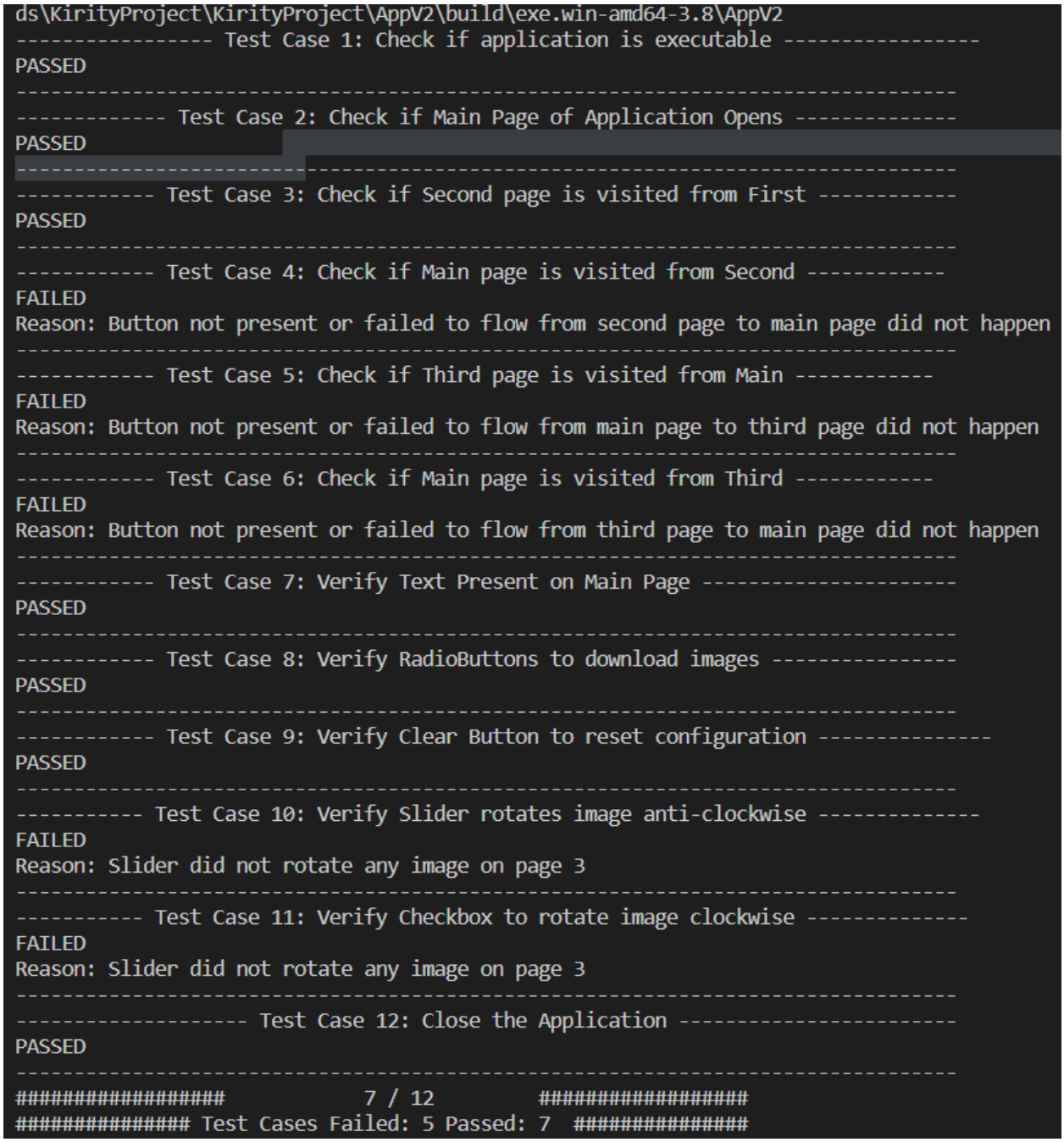
Screenshot:

```
PS C:\Users\ngaur5\Downloads\KirityProject\KirityProject> python .\test_cases_gui.py C:\Users\ngaur5\Downloa
ds\KirityProject\KirityProject\AppV1\build\exe.win-amd64-3.8\AppV1
---------- Test Case 1: Check if application is executable ----------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 2: Check if Main Page of Application Opens ----------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 3: Check if Second page is visited from First ----------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 4: Check if Main page is visited from Second ----------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 5: Check if Third page is visited from Main ----------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 6: Check if Main page is visited from Third ----------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 7: Verify Text Present on Main Page ----------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 8: Verify RadioButtons to download images ----------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 9: Verify Clear Button to reset configuration ----------
PASSED
--------------------------------------------------------------------------------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 11: Verify Checkbox to rotate image clockwise ----------
PASSED
--------------------------------------------------------------------------------
---------- Test Case 12: Close the Application ----------
PASSED
--------------------------------------------------------------------------------
##################           12 / 12         ##################
############### Test Cases Failed: 0 Passed: 12  ###############
```

AppV2 Results:

Command: `~ python test_cases_gui.py <complete absolute path of the application>/AppV2`

Screenshot:



Review Report:

As expected all the test cases were approved for the first GUI application (AppV1), but for the second GUI application (AppV2) only 7 of 12 test cases were approved and the rest 5 failed as seen in above screenshots. Reasons to fail the test cases are as follows:

- Test Case 4: Check if the Main page is visited from Second - FAILED: This test case failed because there were no buttons present in the main page to navigate to other respected pages.
- Test Case 5: Check if Third page is visited from Main - FAILED: This test case also failed because there were no buttons present in the main page to navigate to other respected pages.
- Test Case 6: Check if Main page is visited from Third - FAILED: This test case failed because there were no buttons present in the main page to navigate to other respected pages.
- Test Case 10: Verify Slider rotates image anti-clockwise - FAILED This test case failed because the slider was horizontal for this app and vertical for the original previous app.
- Test Case 11: Verify Checkbox to rotate image clockwise - FAILED This test case also failed because the slider was horizontal for this app and vertical for the original previous app.

Tool Assessment

Set of features and functionalities provided:
PyAutoGUI provides a set of features and functionalities to automate GUI testing, such as GUI element identification, GUI navigation, GUI actions like mouse clicks and keyboard presses, GUI image recognition, and GUI automation through script creation. However, it may not provide advanced testing features like test case management, test reporting, and test scheduling.

Type of coverage:
PyAutoGUI can be used to test various types of GUI elements like buttons, sliders, radio buttons, checkboxes, textboxes, menus, etc. It also supports testing for various platforms like Windows, Linux, and macOS. However, it may not provide coverage for some advanced GUI elements like tree view, data grids, and custom controls.

Reuse of test cases:
PyAutoGUI allows the reuse of test cases, and users can create test cases by recording the interactions with the GUI or by scripting the test cases. The test cases can be easily modified to test different scenarios and can be executed on different machines.

Test results produced:
PyAutoGUI produces test results in the form of console outputs, logs, and screenshots. The test results can be analyzed to identify the failed test cases and the reasons for the failures.

Ease of usage:
PyAutoGUI is easy to use, and users can quickly learn the basics of the tool. However, scripting the test cases may require some programming skills and may not be user-friendly for non-technical users.

Type of GUI elements that can be tested:

PyAutoGUI can be used to test various types of GUI elements, as mentioned earlier. It also provides support for testing web applications through Selenium WebDriver integration. However, it may not provide support for testing mobile applications or desktop applications built on some specific frameworks.

References

[1] OpenCv Library: opencv-python · PyPI
[2] Pyperclip Library: pyperclip · PyPI
[3] PyAutoGui Library: PyAutoGUI · PyPI
[4] Tkinter Library: tkinter — Python interface to Tcl/Tk — Python 3.11.3 documentation
[5] Python GUI Programming With Tkinter: Python GUI Programming With Tkinter – Real Python
[6] Automate UI Testing with PyAutoGUI in Python: Automate UI Testing with PyAutoGUI in Python | by Costas Andreou | Towards Data Science