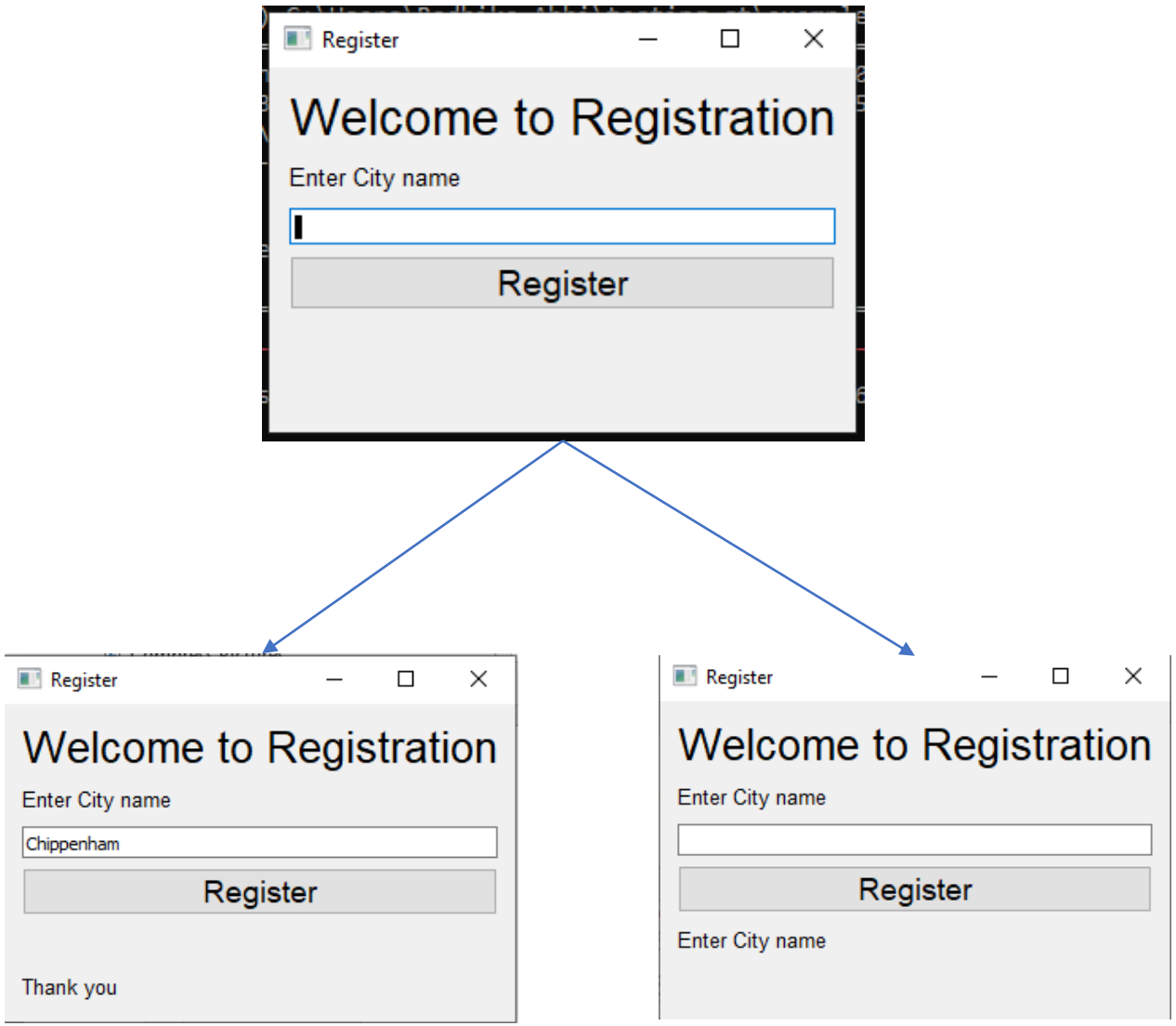
Code & Explanation

Requirement

- Simple application to register user to a city. User enters a city name and taps button, then receives onscreen confirmation or error dependant on text field validation.
  - App name = Register

  Elements:
  - Static text - **welcomeMessage** - “Welcome to Registration”
  - Static text - **enterCity** - “Enter city name”
  - Static text - **cityError** - “Please enter city”
  - Static text - **confirmation** - “Thanks”
  - Text field - **city**
  - Button - **registerButton** - “Register”

UI under Test Register.py



- Register.py is the Application developed as per the requirement on Slide 2.
- The code explanation for GUI is explained from slides 3-5

Code Explanation for App under test "Register.py"

```
import sys
from PyQt5.QtGui import QFont
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout

class CityRegApp(QWidget):

    def __init__(self):
        super().__init__()
        # Create static text for welcomeMessage
        self.welcomeMessage = QLabel()
        self.welcomeMessage.setText("Welcome to Registration")
        self.welcomeMessage.setFont(QFont('Arial', 20))

        #Create static text for display message
        self.enterCity = QLabel()
        self.enterCity.setText("Enter City name")
        self.enterCity.setFont(QFont('Arial', 10))

        #Create text box for Entering City Name
        self.city = QLineEdit()
        self.city.move(20,20)
        self.city.resize(280,40)

        #Create Push Button
        self.registerButton = QPushButton("Register")
        self.registerButton.setFont(QFont('Arial', 14))

        #Create a Static text box for Displaying Error Message
        self.cityError = QLabel()
        self.cityError.setText("")
        self.cityError.setFont(QFont('Arial',10))
```

Create static text for welcomeMessage

Create static text for informing user to enter City Name

Create Editable field for Entering City Name

Create static text for Entering City Name

Create static text for displaying error message

Code Explanation for App under test "Register.py"

```
        #Create a function on button click for next steps
        self.registerButton.clicked.connect(lambda: self.CheckCity())

        # set UI size
        self.setGeometry(50, 50, 300, 200)
        # Set UI Name
        self.setWindowTitle("Register")

        self.layout = QVBoxLayout(self)
        self.layout.addWidget(self.welcomeMessage) # Add Welcome Message to Layout
        self.layout.addWidget(self.enterCity)       # Add Enter City Name to Layout
        self.layout.addWidget(self.city)            # Add Text box to Layout
        self.layout.addWidget(self.registerButton)  # Add Register button to Layout
        self.layout.addWidget(self.cityError)       # Add Static Text for cityError (Default is empty)
        self.layout.addWidget(self.confirmation)    # Add Static Text for CityConfirmation (Default is empty)
        self.setLayout(self.layout)

    def CheckCity(self):
        cityname = self.city.text()         # Read City name entered by user
        self.cityError.clear()              # Clear the old entries if any from the cityError field
        self.confirmation.clear()           # Clear the old entries if any from the confirmation field
        if len(cityname)>0:                 # Checking if the name entered, should not be empty
            self.confirmation.setText("Thank you")      # if not empty display Thank you
        else:
            self.cityError.setText("Enter City name")   # if empty display Enter City name

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = CityRegApp()
    ex.show()
    sys.exit(app.exec_())
```

Create pushbutton backend function for validating user entry

Set UI size & Geometry

Add the created widgets to the Layout

Function body for CheckCity() to validate user input.

Main function using App.exec_()

Assignment

Please refer test_Automation.py for automation code

- Write methods to:

- Test that all elements are displayed on screen, with the exception of cityError and confirmation. Include in test that cityError is NOT displayed. ***(Test Case 1)***

- Test that Thank you message is displayed once a city name is entered and Register button is tapped ***(Test Case 2)***

- Test that when city name is blank, when registerButton is tapped then cityError is displayed ***(Test Case 3)***

Test Approach

- Access the GUI and its structure fields
- Identify the fields for evaluation for each Test case
  - TC1 fields for evaluation (welcomeMessage , enterCity, city, registerButton)
  - TC2 fields for evaluation (confirmation)
  - TC3 fields for evaluation (cityError)
- Check the fields for evaluation in negative state as well
  - TC1 Confimration & City error should not to be displayed or should read as blank
  - TC2 cityError should not be displayed
  - TC3 confirmation field should not displayed
- Setup the test environment (venv)
- Access the app instance in Test script using qtbot
- Use the assert statement on identified text fields

Value add, since the requirement to test for special character and number was not mentioned this was not implemented but it can be checked with below statement
from string import ascii_letters, digits

set(text).difference(ascii_letters + digits)

Refer slide 11 for Automation Test code explanation

Test Case 1

Test that all elements are displayed on screen, with the exception of city error and confirmation. Include in test that cityError is not displayed.

Pre condition:
Setup virtual Environment & Launch the App using the commands mentioned below

Launch command window
python -m venv testing_qt
cd .\testing_qt\
.\Scripts\activate
mkdir example_qt (not required on rerun)
cd example_qt
py .\Register.py

Testrail Fields

Test Steps

- Launch the application using the command
- Verify the following fields in application are displayed
  - welcomeMessage-→"Welcome to Registration" (Static text)
  - enterCity →"Enter city name" (Static text)
  - registerButton → "Register" (Button)
  - City→" " (Blank field) (EditLine)
- Verify the following fields are not displayed
  - confimration → "Thanks" (Static text)
  - cityError → "Please enter city" (Static text)

Expected Results

- Application window is launched and visible
- User is able to see the fields & the button as mentioned in Test Steps descriptions
- User is not the able to see the fields as mentioned in the Test steps descriptions

PASS /FAIL

PASS/FAIL

PASS/FAIL

pytest .\test_Register.py → *to launch automated script for verification*

Test Case 2

Test that Thank you message is displayed once a city name is entered and Register button is tapped

Pre condition:
Setup virtual Environment & Launch the App using the commands mentioned below

Launch command window
python -m venv testing_qt
cd .\testing_qt\
.\Scripts\activate
mkdir example_qt (not required on rerun)
cd example_qt
py .\Register.py
pytest .\test_Register.py

Test Steps

- Launch the application using the command
- Verify the following fields in application are displayed
  - welcomeMessage-→"Welcome to Registration" (Static text)
  - enterCity →"Enter city name" (Static text)
  - registerButton → "Register" (Button)
  - City→" " (Blank field) (EditLine)
- Verify that user is able to enter text (city name) & Pres Register button
  - City→" " (Blank field) (EditLine)
- Verify that CityError field is not visible

Expected Results

- Application window is launched and visible
- User is able to see the fields & the button as mentioned in Test Steps descriptions
- "Thank you" message is displayed after clicking register button
- "Please Enter city name" should not be visible

PASS /FAIL

PASS/FAIL

PASS/FAIL

PASS/FAIL

Test Case 3

Test that all elements are displayed on screen, with the exception of city error and confirmation. Include in test that cityError is not displayed.

Pre condition:
Setup virtual Environment & Launch the App using the commands mentioned below

Launch command window
python -m venv testing_qt
cd .\testing_qt\
.\Scripts\activate
mkdir example_qt (not required on rerun)
cd example_qt
py .\Register.py
pytest .\test_Register.py

Test Steps	Expected Results	
• Launch the application using the command	• Application window is launched and visible	PASS /FAIL

• Verify the following fields in application are displayed
  • welcomeMessage-→"Welcome to Registration" (Static text)
  • enterCity →"Enter city name" (Static text)
  • registerButton → "Register" (Button)
  • City→" " (Blank field) (EditLine)
	• User is able to see the fields & the button as mentioned in Test Steps descriptions	PASS/FAIL

• Keep the city field (EditLIne) blank and press register button	• User is able to see the cityError field displaying "Enter city name"	PASS/FAIL
• Check that Confirmation field is visible or not	• confirmation field "Thank you" is not displayed	PASS/FAIL

Test Automation Code using PyTest

test_Register.py

```
import pytest
from PyQt5 import QtCore
import Register

@pytest.fixture
def app(qtbot):
    test_Register_app = Register.CityRegApp()
    qtbot.addWidget(test_Register_app)
    return test_Register_app

def test_1_display(app):
    #Testcase 1 : Test that all elements are displayed on screen, with the exception of
    #            cityError and confirmation. Include in test that cityError  is NOT displayed.
    assert app.welcomeMessage.text() == "Welcome to Registration", print("Static text: 'Welcome to Registration displayed' does not match expected output")
    assert app.enterCity.text() == "Enter City name", print("Static text: 'Enter City name' does not match expected output")
    assert app.city.text() == "", print("Text field: is not empty when app is launched")
    assert app.registerButton.text() == "Register", print("Button: Register is not displayed")
    assert app.cityError.text() == "", print("Static text:'City Error' Field shall not be displayed ")
    assert app.confirmation.text() == "", print("Static text: 'Confirmation' shall not displayed")

def test_2_display(app, qtbot):
    #Testcase 2 : Test that Thank you message is displayed once a city name is entered and Register button is tapped
    app.city.setText("Chippenham") # Enter City name in EditField
    qtbot.mouseClick(app.registerButton, QtCore.Qt.LeftButton)
    assert app.confirmation.text() == "Thank you", print("Static text:'City name' is not entered")
    assert app.cityError.text() == "", print("Static text:'' Field is not displayed ")

def test_3_display(app, qtbot):
    #Testcase 2 : Test that Thank you message is displayed once a city name is entered and Register button is tapped
    qtbot.mouseClick(app.registerButton, QtCore.Qt.LeftButton)
    assert app.confirmation.text() == "", print("Static text:'confirmation' Field is not empty")
    assert app.cityError.text() == "Enter City name", print("Static text:'Enter City name' Field is not displayed ")
```

Positive checks

Negative checks

Positive checks

Negative checks

Positive checks

Negative checks

Import necessary libraries
Import App which being tested
And access via qtbot

Test Case 1:
Checks using assert statement for the positive & negative checks
If assertion fails then print message informs the user about reason of failure

Test Case 2:
Opens App and enters city name “Chippenham” and simulates mouse clicks using qtbot.mouseClick()
Using assert checks for positive and negative part of the test case
If assertion fails then print message informs the user about reason of failure

Test Case 3:
Opens App and leaves city name as blank and simulates mouse clicks using qtbot.mouseClick()
Using assert checks for positive and negative part of the test case
If assertion fails then print message informs the user about reason of failure

Results of execution

- All Test case Pass as expected
- Set all values as expected in “Register.py”
- execute using pytest test_Register.py in virtual environment

Register.py

Code file for app under test

Test_Register.py

Automation code for app under test

```
(testing_qt) D:\Test_App\Register_Env\testing_qt\RegisterApp>pytest test_Register.py
============================================================ test session starts ============================================================
platform win32 -- Python 3.9.1, pytest-6.2.2, py-1.10.0, pluggy-0.13.1
PyQt5 5.15.3 -- Qt runtime 5.15.2 -- Qt compiled 5.15.2
rootdir: D:\Test_App\Register_Env\testing_qt\RegisterApp
plugins: qt-3.3.0
collected 3 items

test_Register.py ...                                                                                                                  [100%]

============================================================= 3 passed in 0.54s =============================================================

(testing_qt) D:\Test_App\Register_Env\testing_qt\RegisterApp>
```

Results of execution

- First Test Case failed with change in Welcome message
- Change value of welcome message on line 26 of “Register.py”

  `self.welcomeMessage.setText("Welcome to Registration!")`
- execute using pytest test_Register.py in virtual environment do not change the “Test_Register.py”

Register.py	Test_Register.py
Code file for app under test	Automation code for app under test

The mismatch in welcome message is detected by the test_register.py

```
(testing_qt) D:\Test_App\Register_Env\testing_qt\RegisterApp>pytest test_Register.py
============================================================ test session starts ============================================================
platform win32 -- Python 3.9.1, pytest-6.2.2, py-1.10.0, pluggy-0.13.1
PyQt5 5.15.3 -- Qt runtime 5.15.2 -- Qt compiled 5.15.2
rootdir: D:\Test_App\Register_Env\testing_qt\RegisterApp
plugins: qt-3.3.0
collected 3 items

test_Register.py F..                                                                                                                  [100%]

================================================================= FAILURES ==================================================================
_____________________________________________________________ test_1_display _____________________________________________________________

app = <Register.CityRegApp object at 0x0000022DB61414C0>

    def test_1_display(app):
        #Testcase 1 : Test that all elements are displayed on screen, with the exception of
        #             cityError and confirmation. Include in test that cityError is NOT displayed.
>       assert app.welcomeMessage.text() == "Welcome to Registration", print("Static text: 'Welcome to Registration displayed' doe
s not match expected output")
E       AssertionError: None
E       assert 'Welcome to Registration!' == 'Welcome to Registration'
E         - Welcome to Registration
E         + Welcome to Registration!
E         ?                         +

test_Register.py:15: AssertionError
------------------------------------------------------------ Captured stdout call ------------------------------------------------------------
Static text: 'Welcome to Registration displayed' does not match expected output
========================================================== short test summary info ==========================================================
FAILED test_Register.py::test_1_display - AssertionError: None
======================================================== 1 failed, 2 passed in 0.98s ========================================================

(testing_qt) D:\Test_App\Register_Env\testing_qt\RegisterApp>
```

Results of execution

- Second Test Case failed with change in Welcome message
- Change value of welcome message on line 75 of “Register.py”
  `self.confirmation.setText(“Thanks”)`
- execute using pytest test_Register.py in virtual environment do not change the “Test_Register.py”

Register.py	Test_Register.py
Code file for app under test	Automation code for app under test

The mismatch in welcome message is detected by the test_register.py

```
(testing_qt) D:\Test_App\Register_Env\testing_qt\RegisterApp>pytest test_Register.py
============================================= test session starts =============================================
platform win32 -- Python 3.9.1, pytest-6.2.2, py-1.10.0, pluggy-0.13.1
PyQt5 5.15.3 -- Qt runtime 5.15.2 -- Qt compiled 5.15.2
rootdir: D:\Test_App\Register_Env\testing_qt\RegisterApp
plugins: qt-3.3.0
collected 3 items

test_Register.py .F.                                                                                  [100%]

================================================== FAILURES ===================================================
_____________________________________________ test_2_display _____________________________________________

app = <Register.CityRegApp object at 0x000001A217C81A60>, qtbot = <pytestqt.qtbot.QtBot object at 0x000001A21AB59130>

    def test_2_display(app, qtbot):

        #Testcase 2 : Test that Thank you message is displayed once a city name is entered
        #             and Register button is tapped
        app.city.setText("Chippenham") # Enter City name in EditField
        qtbot.mouseClick(app.registerButton, QtCore.Qt.LeftButton)
        #qtbot.waitUntil(lambda: app.hasFocus())
>       assert app.confirmation.text() == "Thank you", print("Static text:'City name' is not entered")
E       AssertionError: None
E       assert 'Thanks' == 'Thank you'
E         - Thank you
E         + Thanks

test_Register.py:29: AssertionError
---------------------------------------------- Captured stdout call ----------------------------------------------
Static text:'City name' is not entered
============================================ short test summary info ============================================
FAILED test_Register.py::test_2_display - AssertionError: None
========================================== 1 failed, 2 passed in 0.68s ==========================================
```

Results of execution

- Second Test Case failed with change in Welcome message
- Change value of welcome message on line 77 of “Register.py”

  `self.cityError.setText("City name ?")`
- execute using pytest test_Register.py in virtual environment do not change the “Test_Register.py”

Register.py	Test_Register.py
Code file for app under test	Automation code for app under test

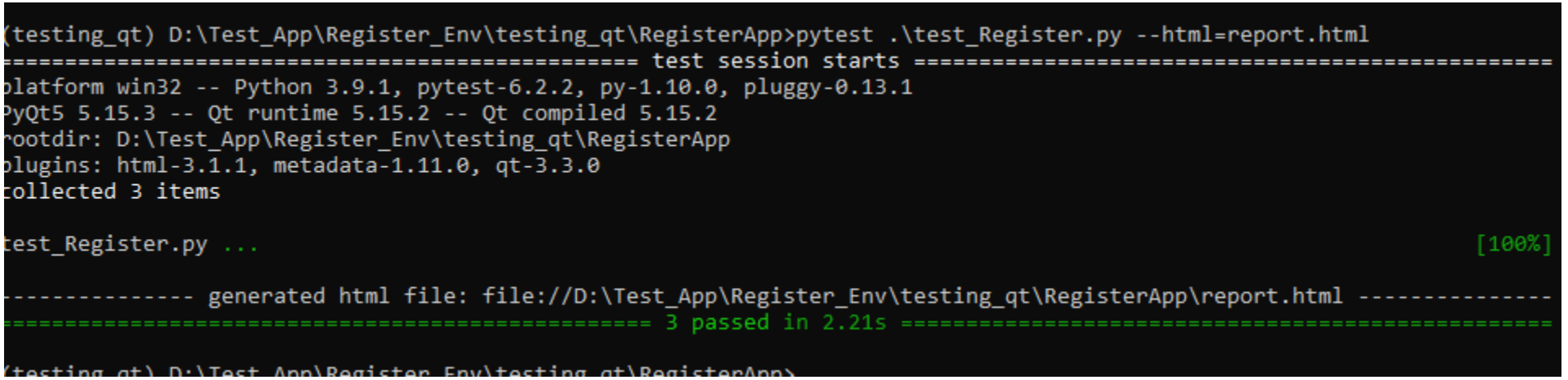
The mismatch in welcome message is detected by the test_register.py

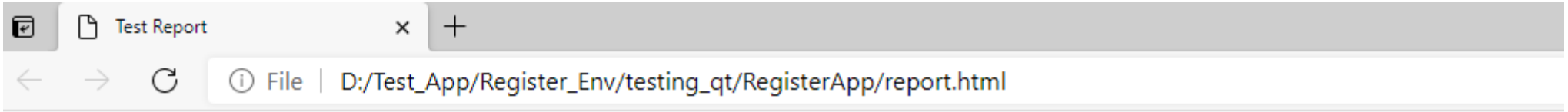
```
(testing_qt) D:\Test_App\Register_Env\testing_qt\RegisterApp>pytest test_Register.py
============================================================ test session starts ============================================================
platform win32 -- Python 3.9.1, pytest-6.2.2, py-1.10.0, pluggy-0.13.1
PyQt5 5.15.3 -- Qt runtime 5.15.2 -- Qt compiled 5.15.2
rootdir: D:\Test_App\Register_Env\testing_qt\RegisterApp
plugins: qt-3.3.0
collected 3 items

test_Register.py ..F                                                                                                                  [100%]

================================================================= FAILURES ==================================================================
_____________________________________________________________ test_3_display _____________________________________________________________

app = <Register.CityRegApp object at 0x000001648ED61F70>, qtbot = <pytestqt.qtbot.QtBot object at 0x0000016491C393A0>

    def test_3_display(app, qtbot):
        #Testcase 2 : Test that Thank you message is displayed once a city name is entered
        #             and Register button is tapped
        qtbot.mouseClick(app.registerButton, QtCore.Qt.LeftButton)
        assert app.confirmation.text() == "", print("Static text:'confirmation' Field is not empty")
>       assert app.cityError.text() == "Enter City name", print("Static text:'Enter City name' Field is not displayed ")
E       AssertionError: None
E       assert 'City name ?' == 'Enter City name'
E         - Enter City name
E         + City name ?

test_Register.py:38: AssertionError
---------------------------------------------------------- Captured stdout call ----------------------------------------------------------
Static text:'Enter City name' Field is not displayed
========================================================== short test summary info ==========================================================
FAILED test_Register.py::test_3_display - AssertionError: None
======================================================== 1 failed, 2 passed in 0.69s ========================================================
```

Added HTML report generation for Pytest

Command to install is pip install pytest-html

Command to execute test is **pytest .\test_Register.py --html=report.html**





report.html

Report generated on 09-Mar-2021 at 11:30:15 by pytest-html v3.1.1

Environment

Packages	{"pluggy": "0.13.1", "py": "1.10.0", "pytest": "6.2.2"}
Platform	Windows-10-10.0.19041-SP0
Plugins	{"html": "3.1.1", "metadata": "1.11.0", "qt": "3.3.0"}
Python	3.9.1

Summary

3 tests ran in 2.18 seconds.

(Un)check the boxes to filter the results.

3 passed, 0 skipped, 0 failed, 0 errors, 0 expected failures, 0 unexpected passes

Results

Show all details / Hide all details

Result	Test	Duration
Passed *(show details)*	test_Register.py::test_1_display	0.59
Passed *(show details)*	test_Register.py::test_2_display	1.49
Passed *(show details)*	test_Register.py::test_3_display	0.00