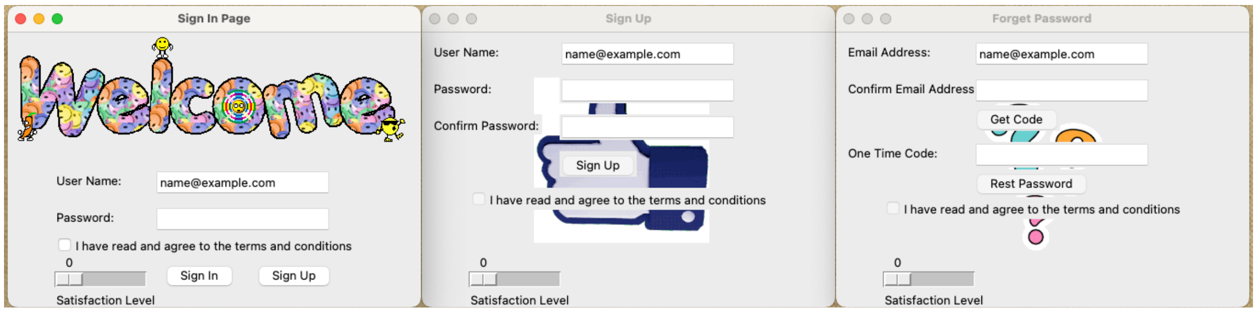
Project 3: Graphical User Interface Testing

Lucy Zhang

Ira A. Fulton Schools of Engineering, Arizona State University
lzhan370@asu.edu

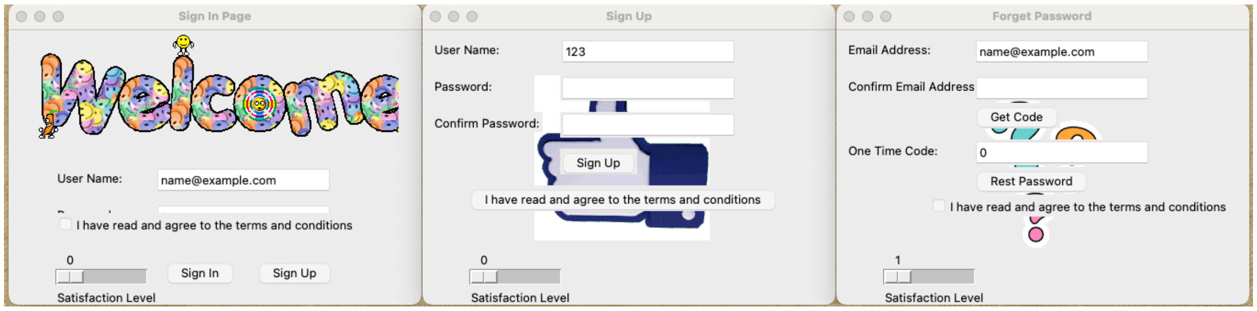
Description of the Application



(Version 1 screenshot)

This application is a simple Graphical User Interface (GUI) application using the tkinter module in Python. This application is designed for user authentication processes, encompassing features like sign-in, sign-up, and password reset functionalities.

- Sign In Page: At the top, there is a welcoming image to enhance user experience. User can enter their User Name and Password to sign in. A checkbox with the label "I have read and agree to the terms and conditions" allows users to acknowledge the terms. A horizontal slider labeled "Satisfaction Level" allows users to indicate their satisfaction level, ranging from 0 to 100. There are two buttons available: Sign In: It triggers the usr_signin function which checks if the entered username and password match any stored in a file. If they do, a welcome message pops up. If the password is wrong, the user gets an option to reset it. If the username isn't found, the user gets the option to sign up and redirects users to the sign-up page.
- Sign Up Page: A canvas displays an image in background. Users can input their desired User Name, Password, and a Confirm Password field ensures they type it correctly. A checkbox for terms and conditions is present. A satisfaction slider similar to the one on the sign-in page. A Sign Up button confirms user registration. If registration is successful, user data is saved.
- Forget Password Page: A canvas displays a question image in background. Users need to provide their email address. A One Time Code field allows users to enter a code for verification. Another checkbox for terms and conditions. A satisfaction slider. Two buttons are present: Get Code: Presumably to get the one-time password/code. Reset Password: To confirm the password reset.



User Name: name@example.com

I have read and agree to the terms and conditions

0

Satisfaction Level

User Name: 123

Password:

Confirm Password:

I have read and agree to the terms and conditions

0

Satisfaction Level

Email Address: name@example.com

Confirm Email Address

One Time Code: 0

I have read and agree to the terms and conditions

1

Satisfaction Level

(Version 2 screenshot)

Version 2 of this GUI application has few changes based on Version 1, such as canvas size, location of button, page flow, scale on satisfaction etc. Every page has three changes in Version 2. The Sign Up botton in Sign In Page opens up Forget Password page instead of Sign Up page.

Description of Unittest

Unittest is a built-in Python library for testing individual units of software to determine if they function correctly.

Features of unittest: Test Discovery:

- Unittest can automatically discover and run test methods.
- Test Fixtures: With unittest, you can set up fixtures using methods such as setUp and tearDown. This allows you to create pre-conditions and post-conditions for your tests.
- Test Suites: Group multiple tests into collections, ensuring that even in large projects, you can select specific subsets of tests to run.
- Assertion Methods: Offers a variety of methods to check conditions, like assertEqual, assertTrue, and assertFalse.
- Mocking and Patching: With unittest.mock, you can mock objects and test methods in isolation without calling the real implementation. This is especially handy for testing user interaction in GUIs.

Scope and Areas of Usage for this GUI Application:

- User Authentication Logic: Test if the application correctly validates existing users. Test if it correctly handles non-existent users and offers them a chance to sign up. Test if it appropriately manages incorrect passwords.
- User Registration Logic: Ensure that passwords and their confirmations match. Ensure new user data is stored correctly in the pickle file. Test the logic that checks for already-existing usernames.
- Password Reset Logic: Although not fully implemented, one could test the logic for sending a one-time code and the process for resetting passwords.
- GUI Component Interactions: Using mocking, simulate user interactions with the GUI components such as button clicks and entry submissions.

- Data Storage: Test the pickling and unpickling process for data storage. Ensure that user data can be read and written without errors.
- Error Messages and Prompts: Ensure that the application throws appropriate error messages under various conditions, such as mismatched passwords or existing usernames.

Description of the Test Cases Developed

Test Main Page:

1. test_main_name_takes_string: Verifies if the main page's username entry is a string.
2. test_var_pwd_takes_string: Checks if the main page's password entry is a string.
3. test_c1_is_checkbox: Validates that the widget c1 on the main page is a checkbox.
4. test_sat_slider_is_slider: Asserts that sat_slider on the main page is a slider widget.
5. test_main_slider_set_value: Ensures that the slider's value on the main page can be set and retrieved correctly.
6. test_canvas_size: Verifies the size of the canvas on the main page.
7. test_c1_location: Checks the location of the c1 checkbox on the main page.

Test Signup Page:

1. test_signup_name_takes_string: Validates if the signup page's username entry is a string.
2. Test_signup_pwd_takes_string: Ensures that the signup page's password entry is a string.
3. Test_c2_is_checkbox: Checks if the c2 widget on the signup page is a checkbox.
4. Test_signup_slider_is_slider: Validates that the signup_slider on the signup page is a slider widget.
5. Test_signup_slider_set_value: Tests if the slider's value on the signup page can be adjusted and retrieved correctly.
6. Test_signup_canvas_size: Checks the size of the canvas on the signup page.
7. Test_c2_location: Validates the location of the c2 checkbox on the signup page.

Test Forget Page:

1. test_forget_name_takes_string: Ensures the forget page's username entry is a string.
2. test_otp_code_takes_string: Validates that the OTP code entry on the forget page is a string.
3. test_c3_is_checkbox: Checks if the c3 widget on the forget page is a checkbox.
4. test_forget_slider_is_slider: Confirms that the forget_slider on the forget page is a slider widget.
5. test_forget_slider_set_value: Tests if the slider's value on the forget page can be adjusted and retrieved correctly.
6. test_forget_canvas_size: Validates the size of the canvas on the forget page.
7. test_c3_location: Checks the location of the c3 checkbox on the forget page.

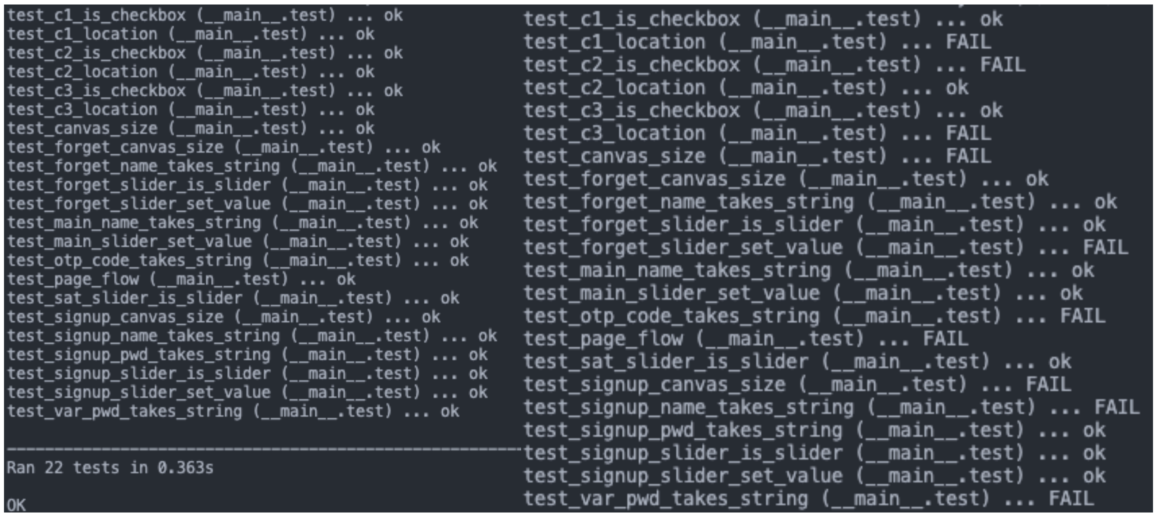
Test Page Flow:

1. test_page_flow: Asserts that the page flow proceeds correctly when the signup button is invoked, ensuring that the signup page is appropriately displayed.

Test	Page	Changes	Version 1	Version 2	Result
1	Main	Var_name	Take string	Unchange	Pass
2	Main	Var_pwd	Take string	Take integer	Pass
3	Main	C1 Element	Checkbox	Unchange	Pass
4	Main	Sat_Slider	Scale 0-100	Unchange	Pass
5	Main	Sat_slider	Slider	Unchange	Pass
6	Main	Canvas	Width = 500	Width = 400	Pass
7	Main	C1 Location	x = 50, y = 220	x = 50, y = 200	Pass
8	Page Flow	Click on Sign Up	Flow to Sign Up	Flow to Forget Password	Pass
9	Sign Up	Signup_canvas	Height = 300	Height = 256	Pass
10	Sign Up	Signup_name	Take string	Take integer	Pass
11	Sign Up	C2 Element	Checkbox	Button	Pass
12	Sign Up	C2 Location	x = 50, y = 170	Unchange	Pass
13	Sign Up	Signup_slider	Scale 0-100	Unchange	Pass
14	Sign Up	Signup_pwd	Take string	Unchange	Pass
15	Sign Up	Signup_slider	Slider	Unchange	Pass
16	Forget Password	Otp_code	String	Integer	Pass
17	Forget Password	Forget_slider	Scale 0-100	Scale 1-100	Pass
18	Forget Password	Forget_name	Take string	Unchange	Pass
19	Forget Password	C3 Element	Checkbox	Unchange	Pass
20	Forget Password	Forget_slider	Slider	Unchange	Pass
21	Forget Password	Forget_canvas	Height = 300	Unchange	Pass
22	Forget Password	C3 Location	x = 50, y = 180	x = 100, y = 180	Pass

Explanation of the Test Results

All test cases passed successfully. The tests covered multiple aspects of the application, spanning three main pages: Main, Sign Up, and Forget Password. The evaluations checked both the integrity of data types and the layout changes between two versions. All these alterations, along with various others across the different pages, were identified and successfully passed their respective test scenarios, confirming Unittest is a useful tool for testing Python GUI application.



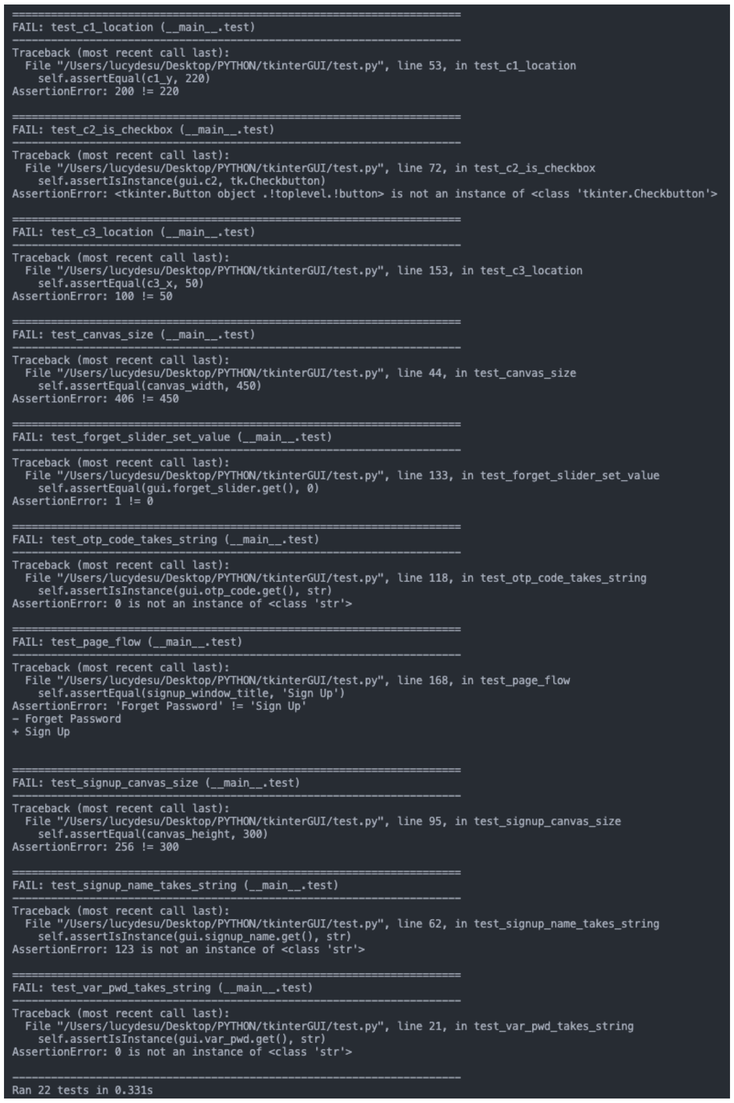
```
test_c1_is_checkbox (__main__.test) ... ok
test_c1_location (__main__.test) ... ok
test_c2_is_checkbox (__main__.test) ... ok
test_c2_location (__main__.test) ... ok
test_c3_is_checkbox (__main__.test) ... ok
test_c3_location (__main__.test) ... ok
test_canvas_size (__main__.test) ... ok
test_forget_canvas_size (__main__.test) ... ok
test_forget_name_takes_string (__main__.test) ... ok
test_forget_slider_is_slider (__main__.test) ... ok
test_forget_slider_set_value (__main__.test) ... ok
test_main_name_takes_string (__main__.test) ... ok
test_main_slider_set_value (__main__.test) ... ok
test_otp_code_takes_string (__main__.test) ... ok
test_page_flow (__main__.test) ... ok
test_sat_slider_is_slider (__main__.test) ... ok
test_signup_canvas_size (__main__.test) ... ok
test_signup_name_takes_string (__main__.test) ... ok
test_signup_pwd_takes_string (__main__.test) ... ok
test_signup_slider_is_slider (__main__.test) ... ok
test_signup_slider_set_value (__main__.test) ... ok
test_var_pwd_takes_string (__main__.test) ... ok

----------------------------------------------------------------------
Ran 22 tests in 0.363s

OK
```

```
test_c1_is_checkbox (__main__.test) ... ok
test_c1_location (__main__.test) ... FAIL
test_c2_is_checkbox (__main__.test) ... FAIL
test_c2_location (__main__.test) ... ok
test_c3_is_checkbox (__main__.test) ... ok
test_c3_location (__main__.test) ... FAIL
test_canvas_size (__main__.test) ... FAIL
test_forget_canvas_size (__main__.test) ... ok
test_forget_name_takes_string (__main__.test) ... ok
test_forget_slider_is_slider (__main__.test) ... ok
test_forget_slider_set_value (__main__.test) ... FAIL
test_main_name_takes_string (__main__.test) ... ok
test_main_slider_set_value (__main__.test) ... ok
test_otp_code_takes_string (__main__.test) ... FAIL
test_page_flow (__main__.test) ... FAIL
test_sat_slider_is_slider (__main__.test) ... ok
test_signup_canvas_size (__main__.test) ... FAIL
test_signup_name_takes_string (__main__.test) ... FAIL
test_signup_pwd_takes_string (__main__.test) ... ok
test_signup_slider_is_slider (__main__.test) ... ok
test_signup_slider_set_value (__main__.test) ... ok
test_var_pwd_takes_string (__main__.test) ... FAIL
```

(Test result from Version 1 and Version 2)



```
======================================================================
FAIL: test_c1_location (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 53, in test_c1_location
    self.assertEqual(c1_y, 220)
AssertionError: 200 != 220

======================================================================
FAIL: test_c2_is_checkbox (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 72, in test_c2_is_checkbox
    self.assertIsInstance(gui.c2, tk.Checkbutton)
AssertionError: <tkinter.Button object .!toplevel.!button> is not an instance of <class 'tkinter.Checkbutton'>

======================================================================
FAIL: test_c3_location (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 153, in test_c3_location
    self.assertEqual(c3_x, 50)
AssertionError: 100 != 50

======================================================================
FAIL: test_canvas_size (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 44, in test_canvas_size
    self.assertEqual(canvas_width, 450)
AssertionError: 406 != 450

======================================================================
FAIL: test_forget_slider_set_value (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 133, in test_forget_slider_set_value
    self.assertEqual(gui.forget_slider.get(), 0)
AssertionError: 1 != 0

======================================================================
FAIL: test_otp_code_takes_string (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 118, in test_otp_code_takes_string
    self.assertIsInstance(gui.otp_code.get(), str)
AssertionError: 0 is not an instance of <class 'str'>

======================================================================
FAIL: test_page_flow (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 168, in test_page_flow
    self.assertEqual(signup_window_title, 'Sign Up')
AssertionError: 'Forget Password' != 'Sign Up'
- Forget Password
+ Sign Up


======================================================================
FAIL: test_signup_canvas_size (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 95, in test_signup_canvas_size
    self.assertEqual(canvas_height, 300)
AssertionError: 256 != 300

======================================================================
FAIL: test_signup_name_takes_string (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 62, in test_signup_name_takes_string
    self.assertIsInstance(gui.signup_name.get(), str)
AssertionError: 123 is not an instance of <class 'str'>

======================================================================
FAIL: test_var_pwd_takes_string (__main__.test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/lucydesu/Desktop/PYTHON/tkinterGUI/test.py", line 21, in test_var_pwd_takes_string
    self.assertIsInstance(gui.var_pwd.get(), str)
AssertionError: 0 is not an instance of <class 'str'>

----------------------------------------------------------------------
Ran 22 tests in 0.331s
```

(Version 2 test result in detail)

Assessment of Unittest

a. Set of Features and Functionalities Provided:

Unittest offers a rich set of assertions to validate output and behavior, like assertIsInstance. It provides setup (setUp) and teardown methods (tearDown) to initialize and finalize preconditions for testing. It's integrated into Python's standard library, ensuring consistent support and updates.

b. Type of Coverage:

The testing here is mainly at the unit level, focusing on individual GUI components' behaviors and properties. The tests cover data type validations, GUI element types, layout attributes like size and position, and application flow.

c. Reuse of Test Cases:

With unittest, test cases encapsulated in functions can be easily reused and rerun. For instance, similar tests for different GUI pages (like the Main, Sign Up, and Forget Password pages) reuse the same testing logic.

d. Test Results Produced:

Results are clearly communicated as 'Pass' or 'Fail'. When integrated with more extensive tools or platforms, detailed logs, error messages, and tracebacks can also be provided for failed tests.

e. Ease of Usage:

unittest follows a straightforward and Pythonic approach, making it easy for developers familiar with Python. With clear function naming and the ability to group related tests, the framework facilitates readability and ease of maintenance.

f. Type of GUI Elements that can be Tested:

With the combination of tkinter and unittest, various GUI elements can be tested: Widgets like Checkbuttons, Sliders, and Canvases. Properties such as size, position, and data types. Functional behaviors like the result of button clicks or page flows. In conclusion, this tool setup provides a comprehensive, easy-to-use environment for testing GUI applications developed with tkinter.