

## Systems Testing

```
import unittest
from alzheimer_app import AlzheimerApp
import tkinter as tk

class TestAlzheimerApp(unittest.TestCase):

    def setUp(self):
        """Set up the application for testing."""
        self.root = tk.Tk()
        self.app = AlzheimerApp(self.root)

    def tearDown(self):
        """Destroy the application after testing."""
        self.root.destroy()

    def test_login_frame_display(self):
        """Test if the login frame is displayed correctly."""
        self.app.show_login()
        login_frame = self.app.login_frame
        self.assertIsNotNone(login_frame, "Login frame should be initialized.")
        self.assertTrue(login_frame.winfo_ismapped(), "Login frame should be visible.")

    def test_login_success(self):
        """Test successful login."""
        self.app.login_view_model = MockLoginViewModel(success=True) # Mocking
successful login
        self.app.username_entry.insert(0, "josh_kay")
        self.app.password_entry.insert(0, "test123")
        self.app.login()
        self.assertIsNotNone(self.app.dashboard_frame, "Dashboard frame should be
displayed after successful login.")

    def test_login_failure(self):
        """Test failed login."""
        self.app.login_view_model = MockLoginViewModel(success=False) # Mocking failed
login
        self.app.username_entry.insert(0, "josh_kay8")
        self.app.password_entry.insert(0, "test1237")
        self.app.login()
        self.assertIsNone(self.app.dashboard_frame, "Dashboard frame should not be
displayed after failed login.")

    def test_dashboard_display(self):
        """Test if the dashboard displays after login."""
        self.app.login_view_model = MockLoginViewModel(success=True)
        self.app.username_entry.insert(0, "josh_kay")
        self.app.password_entry.insert(0, "test123")
        self.app.login()
        self.app.show_dashboard()
        self.assertTrue(self.app.dashboard_frame.winfo_ismapped(), "Dashboard frame
should be visible.")

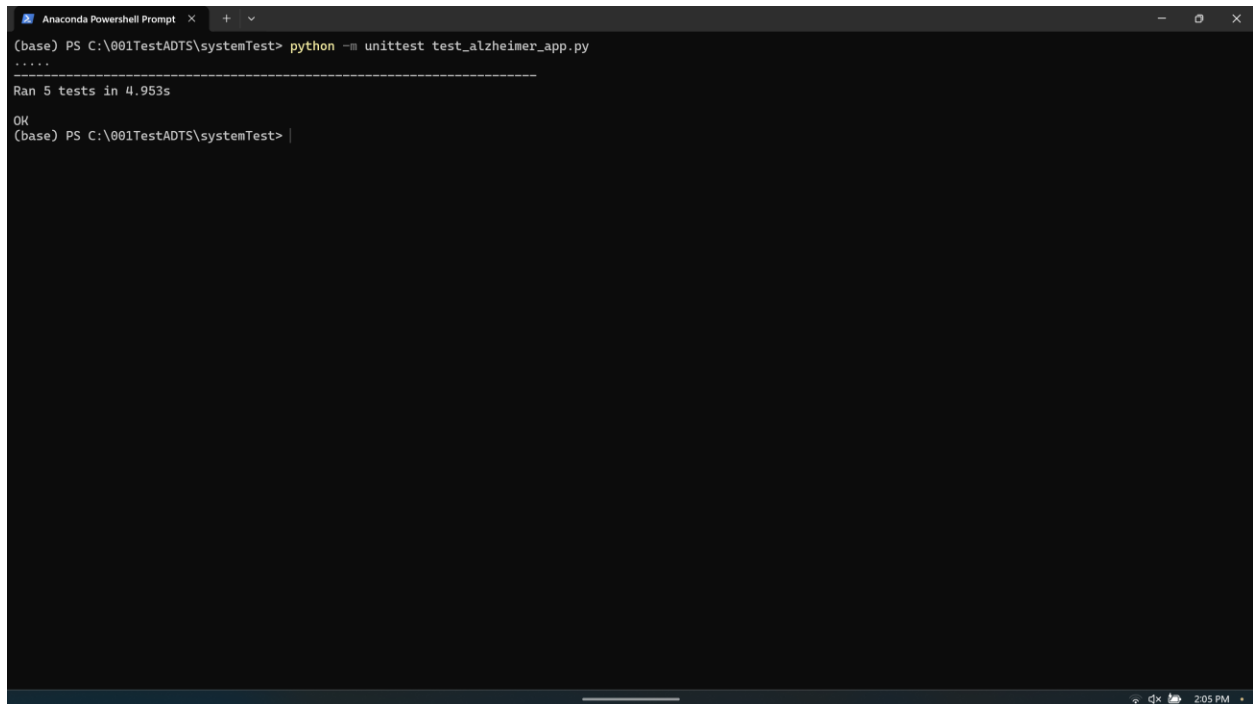
    def test_clear_frame(self):
        """Test if the clear_frame method works correctly."""
```

```
        self.app.show_login()
        self.app.clear_frame()
        self.assertFalse(self.app.login_frame.wininfo_ismapped(), "Login frame should be
cleared.")

class MockLoginViewModel:
    """Mock class for LoginViewModel to simulate login behavior."""
    def __init__(self, success):
        self.success = success

    def login_user(self, username, password):
        return self.success

if __name__ == "__main__":
    unittest.main()
```



The screenshot shows a terminal window titled "Anaconda PowerShell Prompt". The command executed is `python -m unittest test_alzheimer_app.py`. The output indicates that 5 tests were run successfully in 4.953 seconds, resulting in an "OK" status. The terminal window has a dark background with light-colored text. The status bar at the bottom right shows the time as 2:05 PM.

```
Anaconda PowerShell Prompt
(base) PS C:\001TestADTS\systemTest> python -m unittest test_alzheimer_app.py
.....
Ran 5 tests in 4.953s

OK
(base) PS C:\001TestADTS\systemTest> |
```

```
Anaconda Powershell Prompt x + v
(base) PS C:\001TestADTS\systemTest> python -m unittest test_alzheimer_app.py
.....
Ran 5 tests in 4.953s

OK
(base) PS C:\001TestADTS\systemTest> |
```

Show hidden icons

2:06 PM  
11/25/2024